ENHANCING A VIRTUAL DISTRIBUTED LIBRARY USER INTERFACE VIA SERVER-SIDE USER PROFILE CACHING

THESIS

Jason T. Ward, Captain, USAF
AFIT/GCS/ENG/00M-23

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GCS/ENG/00M-23

ENHANCING A VIRTUAL DISTRIBUTED LIBRARY USER INTERFACE VIA SERVER-SIDE USER PROFILE CACHING

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Systems

Jason T. Ward, B.S.

Captain, USAF

March 2000

AFIT/GCS/ENG/00M-23

ENHANCING A VIRTUAL DISTRIBUTED LIBRARY USER INTERFACE VIA SERVER-SIDE USER PROFILE CACHING

Jason T. Ward, B.S.
Captain, USAF

Approved:

Michael L. Talbert, Ph.D., Major, USAF — 6 Mar 2000
Thesis Committee Chairman — Date

Karl S. Mathias, Ph.D., Major, USAF — 8 MAR 2000
Thesis Committee Member — Date

Terry A. Wilson, Ph.D., Captain, USAF — 8 MAR 2000
Thesis Committee Member — Date

Acknowledgments

Many people have contributed their time and expertise to this research effort. I would like to extend my sincere gratitude to everyone who has had played a part in seeing the successful completion of this research endeavor.

First, I would personally like to thank Maj Michael Talbert for his assiduous guidance and ceaseless enthusiasm throughout this research effort. I also would like to acknowledge the contribution of the members of my sponsoring organization, the Air Force Research Laboratory (AFRL), Sensors Directorate. I am sincerely thankful for the combined efforts of Capt Terry Wilson and Mike Brogan for providing me with useful insight into the Virtual Distributed Laboratory and the Advanced Query Tool and providing the excellent hardware/software on which this effort was conducted. I also want to acknowledge Maj Karl Mathias for the key insights he has provided.

For her love and encouragement, I would like to express my grateful and heartfelt appreciation to my wife, Liz. She has shown an enormous amount of patience and understanding throughout the last eighteen months.

Finally, I offer thanks to God, the Creator, who is my true source of inspiration and has blessed me with joy throughout my life.

Table of Contents

Page

Page

List of Figures

Figure Page

List of Tables

Table Page

Abstract

Various Department of Defense (DoD) agencies archive terabytes of intelligence imagery and electrooptical signature data. The Air Force Research Laboratory, Sensors Directorate (AFRL/SN), is tasked with creating and managing a virtual distributed library that facilitates secure, detailed queries across these distributed holdings using the internally developed *Advanced Query Tool* (AQT). In this research, a methodology is proposed to utilize user profiling techniques to augment a digital library. As part of this methodology, product-oriented usability analysis metrics are introduced that quantitatively verify the usability of an interface. The methodology is applied to the AFRL/SN's Virtual Distributed Laboratory AQT and subsequently analyzed using the suite of product-oriented usability metrics. The results of applying the methodology to the test case demonstrated significant, quantitatively verifiable improvements to the AQT interface. This research establishes that user profiling may be utilized to greatly improve the user interface of a digital library.

ENHANCING A VIRTUAL DISTRIBUTED LIBRARY USER INTERFACE VIA SERVER-SIDE USER PROFILE CACHING

1 Introduction

The Department of Defense (DoD) possesses a vast wealth of intelligence imagery and electrooptical target signature data residing in large databases located at geographically-separated government facilities across the nation. The Sensors Directorate (SN) of the Air Force Research Laboratory (AFRL), located at Wright-Patterson Air Force Base in Dayton, Ohio, is tasked with making these terabytes of detailed imagery and signature data available to end-users at diverse locations. AFRL/SN has organized a Virtual Distributed Laboratory (VDL) that employs the World-Wide Web (WWW) to provide anywhere, anytime distributed database access. A web-based interface utilizing browsers and Java™ applets is used to search for these images and retrieve ones that may be appropriate to a user's request. However, two crucial challenges remain in achieving the VDL's objective. The first challenge is the development of a profiling technology that extrapolates and records information about user preferences during interactive data-inquiry sessions. This data is used to

enhance subsequent sessions by reducing the amount of required user interaction for signature data subset specification. The second challenge is the quantitative verification of an augmented query interface designed to increase the interactive directness of the query application.

The remainder of this chapter is devoted to how this research approached these challenges, and it is divided as follows: Section 1.1 describes the problem the VDL faces in detail, Section 1.2 outlines the goals of this research, Section 1.3 gives the approach that was taken, Section 1.4 specifies the scope of this research, and Section 1.5 lays out the organization of this document.

1.1 Problem Statement

The AFRL/SN Virtual Distributed Laboratory is essentially a virtual toolbox for testing and evaluating image processing algorithms using imagery and signature data held by numerous DoD agencies. In addition to the sheer volume of data holdings, many agencies have developed metadata databases for their repositories, describing details of the images themselves. In order to take advantage of these metadata databases, the VDL is tasked with implementing the vision depicted in Figure 1. The figure shows end-users able to remotely query the contents of these metadata databases via the WWW. The first iteration of the application intended to implement this vision was coined the *Advanced Query*

Tool (AQT) *1.0*. However, the AQT 1.0 could not be fielded due to critical issues that had not been foreseen. The AQT 1.0 was not configurable on the AQT server-side—therefore it could not be used in conjunction with any metadata database other than the one at the VDL central host at Wright-Patterson Air

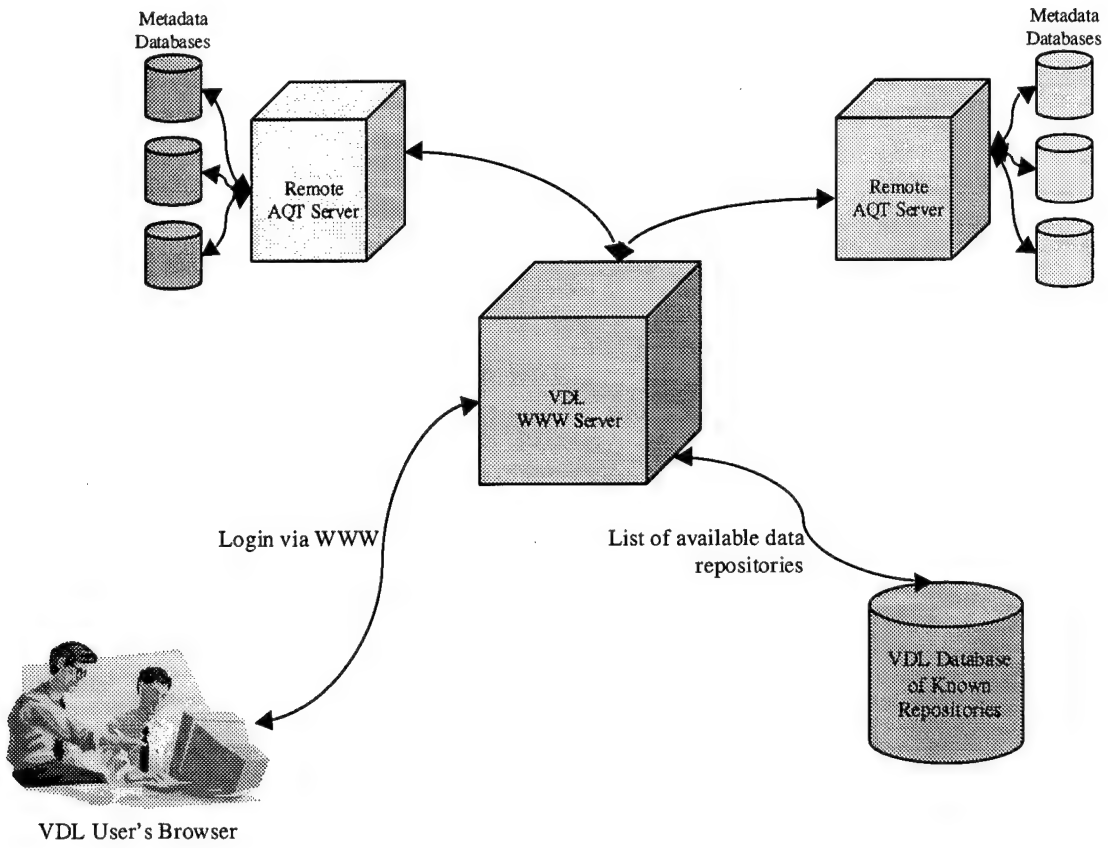

Figure 1: Vision for Future VDL-Access

Force Base. Additionally, no capability was built into the AQT 1.0 for tracking which data attributes were important to the user, hence a useful query required an unacceptable multitude of mouse-clicks to specify.

The first issue was addressed in a new design of the *Advanced Query Tool* (AQT), release 2.0. The AQT 2.0 specification requires that an AQT server should be easily configured to meet the needs of the individual metadata database administrators. Administrators should be able to select any JDBC™-supported database(s) and publish which portions of the database(s) that registered remote users are permitted to query.

However, in order to satisfy the user activity-tracking issue, this research proposes the exploitation of *user profiling*. Incorporating user profiling should allow a system to provide the users with easily generated, quickly recalled, custom-built queries to express their specific data needs more quickly and efficiently. For example, a user's query history might reveal a preponderance of requests for tank imagery in a highly specific spectral region. This user's query selections would default to tank imagery in those regions to accelerate the query process. Furthermore, user profiling enables the central tracking of critical user information (e.g. user identity and privileges), logging of transactions, usage statistics, and ultimately the construction of comprehensive user histories.

In addition to the inclusion of user profiling technology, this research attempts to alleviate another problem—no analysis had been previously performed to ensure the interface's efficiency in terms of usability had actually

increased with the creation of the AQT 2.0.

1.2 Goal

The issues highlighted above enumerate the *fourfold* goal of this research:

1) To evaluate the usability of the AQT 1.0 and AQT 2.0 interfaces using product-oriented usability analysis.

2) To implement user profiling in the AQT 2.0.

3) To gauge the improvement in usability from the AQT 1.0 to the AQT 2.0 modified to include user profiling.

4) To suggest future enhancements involving the exploitation of central storage of metadata queries and user histories for the purpose of escalating VDL-capability.

1.3 Approach

This research effort was conducted in several steps. It commenced with an examination of the current interface and query capability of the AQT 1.0 and AQT 2.0. The next step consisted of a literature review of digital libraries, user profiling techniques, and interface usability metrics. Based on the outcome of this research, applicable techniques were chosen from the literature for experimental purposes. These techniques were utilized to complete an interface

usability metrics-based analysis on the AQT 1.0 and AQT 2.0. This yielded a quantitative comparison in usability terms of the advantages and drawbacks of the improvements to the interface of the AQT 1.0. Moreover, this research proposed and tested the implementation of user profiling in the AQT 2.0.

1.4 Scope

Although AFRL/SN's VDL works with numerous database platforms at the host and outlying locations, for purposes of this research, only the metadata database currently residing at the VDL was used (an Oracle 8*i*™ database). This database serves as a representative sample of the type and quality of available metadata and degree of effort required to integrate it into the AFRL/SN's VDL-vision.

1.5 Document Organization

Chapter 2 introduces digital libraries, user profiling technology, interface usability metrics, and the underlying AFRL/SN VDL architecture, describing the current implementation of image and signature data queries. Chapter 3 details the steps for performing an interface usability analysis and a method for optimizing VDL-capability using user profiling. In Chapter 4, a modified AQT 2.0 is shown that incorporates user profiling. In addition, an evaluation of the interface usability is performed using the VDL's AQT 1.0 and modified AQT 2.0.

Finally, Chapter 5 summarizes the findings, recommendations, and future directions for research in this and related areas.

2 Background

As a precursor to understanding the methodology employed in this research effort, a fundamental appreciation for the issues inherent to the mission and operation of AFRL/SN's Virtual Distributed Laboratory is essential, along with an understanding of user profiling technology. This chapter provides an overview of these issues in four main sections. The first section, 2.1, gives an overview of digital libraries, including their query interfaces. The second part of the chapter, section 2.2, discusses user profiling technology and its applications. In section 2.3 the VDL in detail is analyzed, including its configuration, user interface, and query capability. Section 2.4 introduces product-oriented usability testing metrics for evaluating user interfaces. Finally, section 2.5 provides a summary and presents conclusions based upon the previous sections.

2.1 Overview of Digital Libraries

This section provides an overview of digital libraries. It is broken into three main sections. In section 2.1.1, digital libraries are defined. Section 2.1.2 continues to describe the requirements for digital libraries. In section 2.1.3, query interfaces to digital libraries are discussed and design principles are introduced.

2.1.1 Digital Library Definition

The definition of a digital library is more elusive than one might think.

One method to arrive at the definition of an entity is to describe the activities it undertakes or satisfies. A multitude of buzzwords exists in an attempt to define the activities related to a digital library. These include (but are not limited to): *multimedia database, information mining, information warehouse, information retrieval, on-line information repositories, electronic library, operational image applications,* and *wide area information services* (WAIS) [GLADN96]. However, the number of distinct activities this list represents is not so overwhelming. A large overlap exists between these categories, allowing the digital library, itself, to perform this entire myriad of functions. A more common definition for a digital library is a "distributed technology environment which dramatically reduces barriers to the creation, dissemination, storage, ...and reuse of information by individuals and groups" [FRANK96]. This is a narrower description that highlights the benefits of digital, reusable technology.

2.1.2 Digital Library Requirements

An analysis of digital library requirements performed by IBM [HARVE99] identified several hundred specific needs. A number of broad categories of requirements emerged and are summarized below.

Distribution: People are often distant from information sources, frequently in locations where high-speed links are unavailable. Such users of large files

often have special needs such as delayed delivery times (till less busy times of the day) and smaller file sizes.

Performance: Updating a document or image is likely to be a rare event and not subject to stringent responsiveness. However, retrieval should be rapid, and searching even more optimized. Giving partial research results while undertaking a lengthy search in the background may help satisfy users.

Accessibility: Different users will be running digital library applications on different platforms. This may be because of the history of the user's organization, the application's functionality, or the user's personal preference. Library services must be accessible from various platforms.

Scalability: Digital library service providers want to start small and grow without disruption or breakage. There should be no system-imposed limit to collection sizes.

Overhead: Installation and custodial responsibilities for a digital library should require only a small addition in time and training for administrators. Installation and use of the service portion of the library should be easy and not require help from specialists.

Standardization: The previous requirements imply a long-term

commitment to a specific programming interface for library services and underlying communication services.

2.1.3 Query Interfaces to Digital Libraries

In order to search a digital library, a usable query interface must be provided. This interface may be accomplished through something as simple as a textbox or as complicated as a series of sliders, buttons, and other devices that provide a more dynamic interface [SHNEI94]. No matter what style of interface is developed, several design principles are requisite for effective query interfaces.

Strive for consistency. Terminology, instructions, colors, and fonts should be used consistently throughout the interface. For example, changing the search-initiation button label from "search" to "query" has been shown to slow user performance and lower efficiency scores [MUELL99].

Provide shortcuts for skilled users. An example where this is important is users who already know a term should not have to perform a time-consuming search or navigation through a lengthy series of menus or dialogs to select the term already in-mind.

Offer informative feedback. The user may desire to be informed of all aspects of the search they are undertaking: the sources, fields, and allowable variants.

When the search terminates, not only search results, but optionally the search path or strategy should be available to the advanced user.

Design for closure. Users should know when a search has completed or when they have viewed every item in a browse list. Traversing a deep menu tree is disorienting, especially if backtracking and exploration are expected. Normally, a broader tree with fewer levels is preferred, since it allows users to reach their destination in fewer steps and reduces short-term memory loads for the search engine.

Offer simple error-handling. Syntax errors should be detected and consequently rejected wherever possible—all error messages should be specific, constructive, and no more technical than necessary. For example, an error message stating "eval_query 50: inq_eval_query called with zero length query," is nearly incomprehensible to most non-programmers. A preferable statement would be "No search text was given. Enter text and try again." [MUELL99]

Permit easy reversal of actions. Every action should be reversible so users may return to a previous state during a session. An example would be storing queries given and allowing users to re-issue them. This is particularly valuable if the complete context of the query (e.g. feedback about the relevance of returned items) is also captured.

Support user control. In well-designed interfaces, users initiate an action, monitor its progress, and always feel in control. It is preferable to design interfaces with no enforced sequence of actions—users should be able to set parameters for a query in whichever order they prefer. Another way to give users a sense of control is provide a visual overview of the entire search area. [MUELL99]

2.2 User Profiling Technology

With the continued evolution of Internet communications technology, end user relationships have become the latest focus of successful distributed applications. These applications are responsible for the dissemination of increasingly large amounts of data based upon the leveraging of valuable relationships with end users. Every visit performed by the end user provides an opportunity to capture and better understand that user's needs and desires. Section 2.2 will serve as an overview of how information is captured and leveraged from end user interactions.

2.2.1 Gaining User Knowledge Through Interaction

Every interaction with a user is valuable, allowing the capture of three basic forms of information about the customer: *transaction information, user preferences,* and *facts about the user. Transaction information* consists of information

about which products the user has selected, the amount of information a user has downloaded, and dates and times of usage. *User preferences* reflect user likes and dislikes which may reflect future product selection. Preferences can range from liking the color red to only looking for tank imagery. Finally, *user facts* are specific pieces of information about the user that rarely change and serve to describe the user. Examples include things like names, passwords, usage justifications, user domains, etc. [HARVE99]

2.2.2 *Customization Via Repetition*

In the commercial world, many merchants are learning an important lesson about doing business on the Internet. If one baits his hook with markdowns and waits for shoppers to come to his site, customers may be pulled in but are easily lost when a better offer comes along. It is far better to provide a higher quality level of service and customized offerings to keep customers coming back. Don Peppers, an author of several books on customer relationship management and interactive marketing, says "the secret to keeping customers is to come up with a strategy that makes it in their best interest to remain loyal rather than switch to a competitor" [MUELL99]. He explains that the best way to do that is to build a Learning Relationship. A Learning Relationship is one in which the customer tells the service provider what he wants and receives it from

the provider in the manner specified. Repeating this process enables the seller to anticipate the buyer's needs and customize service incrementally with every customer interaction. The customer is now seen as a core business asset as opposed to an accounting widget [MCKIE99].

All of the logic in the preceding paragraph applies to military applications as well. End users will not continue to use the same applications if other ones exist that provide customized service. An application that is customized for the end user will be perceived as more valuable and provide better application usage in terms of speed and efficiency. The ultimate value behind customization is the accomplishment of the mission to provide the highest level of service to the end user.

2.3 The Virtual Distributed Laboratory (VDL) Architecture

The mission of AFRL/SN's VDL is to support algorithm evaluators, imagery/signature data collectors and users, and Automatic Target Recognition (ATR) researchers throughout the DoD [VDLFO99]. Agencies in the DoD have collected the data in which these users are interested. The problem is these users have no idea who has what data in what quantities and if it is of use to their project. The objective becomes the creation of a query tool to locate data which fits a user's criteria regardless of where that data physically resides within the

DoD. This query tool would then be distributed to the various agencies to allow remote users to query imagery metadata collections by ultimately directing them to the physical source of the data itself.

2.3.1 Advanced Query Tool, Release 1.0

The growing desire for online access to vast imagery/signature data has led to the creation of a tool, by the AFRL/SN Target Recognition branch, to satisfy this objective. The first iteration of it was called the Advanced Query Tool (AQT) 1.0. Figure 2 is a simplified depiction of how it works:

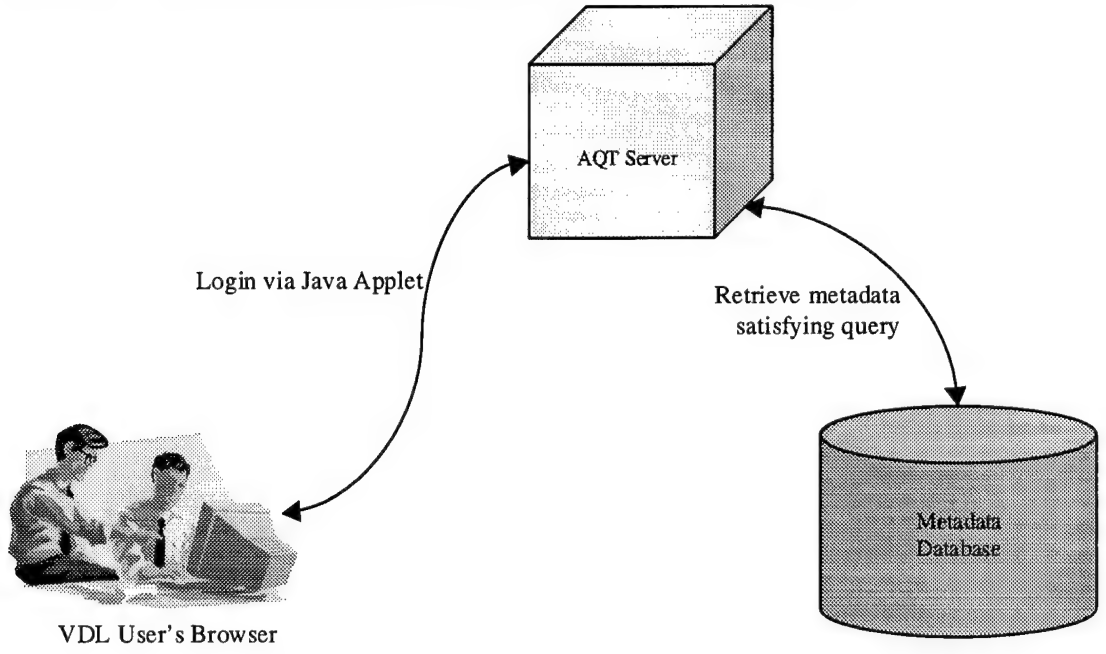

Figure 2: Advanced Query Tool 1.0 Functional Diagram

The VDL user loads a Java™ applet via a web browser and logs into the AQT Server hosted at AFRL/SN's VDL site. The term "login" is almost a

misnomer, because no actual user tracking or verification takes place within the system; security is implemented externally by not allowing unauthorized users access to the web page based upon their IP address and username/password. The AQT 1.0 initial query building screen appears in the browser as in Figure 3.

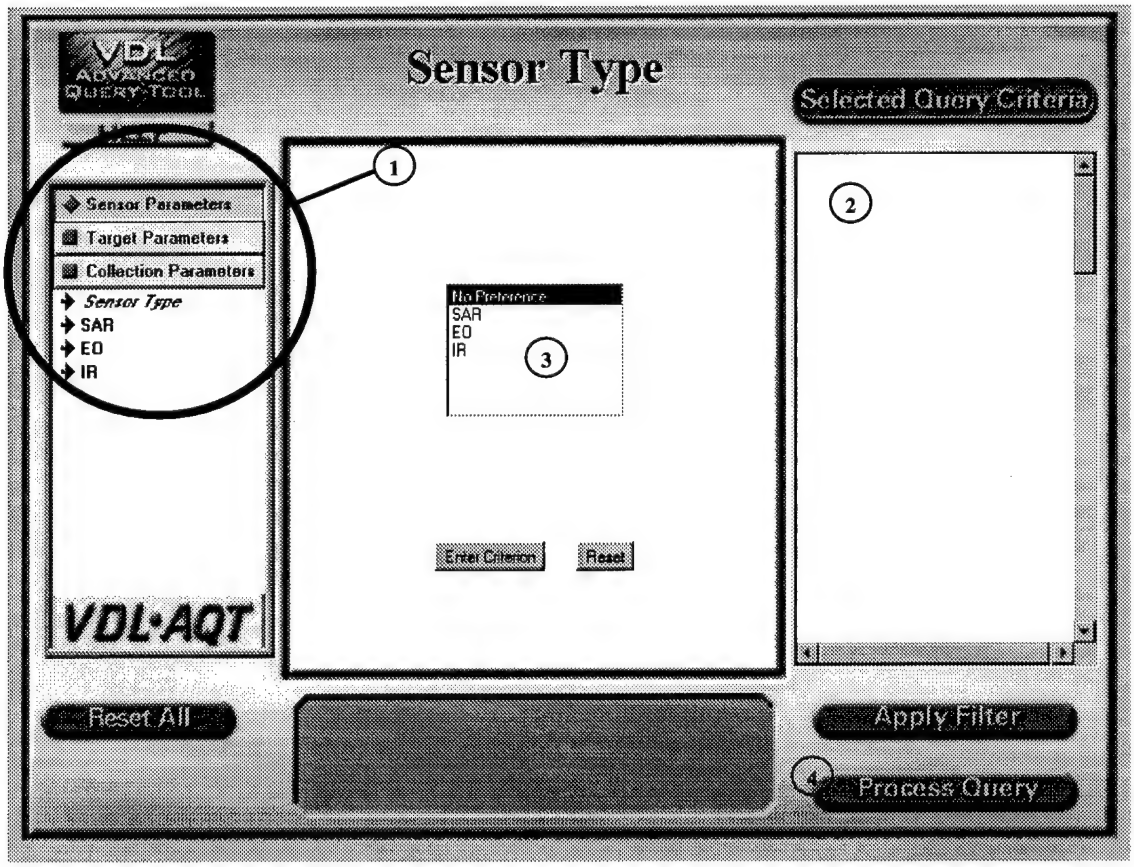

Figure 3: Advanced Query Tool 1.0 Screen Shot

Query criteria are specified using numerous drop-down menus (1). "Sensor Parameters", "Target Parameters", and "Collection Parameters" appear in Figure 3 with "Sensor Parameters" indicated having been selected. Parameter

selections appear in the empty box on the right (2). The actual query is constructed dynamically by selecting values provided for each specified criteria (3). Once a user has finished choosing query criteria, the user selects the "Process Query" button (4). This causes the query to be sent to the AQT Server in Figure 2 for processing. This processing consists of the server retrieving the appropriate metadata from the central metadata server database and sending the results back to the client applet via Hypertext Transfer Protocol (HTTP). At this point, the applet can display the results to the user in one of the following formats: Aspect Coverage, 3D Aspect Coverage, Bar Chart, or Pie Chart, as shown in Figure 4.

However, the AQT 1.0 was more of a proof-of-concept demonstration and was abandoned due to the two main issues highlighted in Section 1.1: 1) The server-side is not configurable to work with any metadata database other than the one at the VDL host, and 2) No capability was built into the AQT 1.0 for capturing complex metadata queries performed by the user to be used for recall and reuse in subsequent sessions.

2.3.2 Advanced Query Tool, Release 2.0

As explained in Section 1.1, the new design of the AQT 2.0 (depicted in Figure 5), included a separate application (the AQT Server Initial Configuration Program) that could configure any given metadata database to operate in

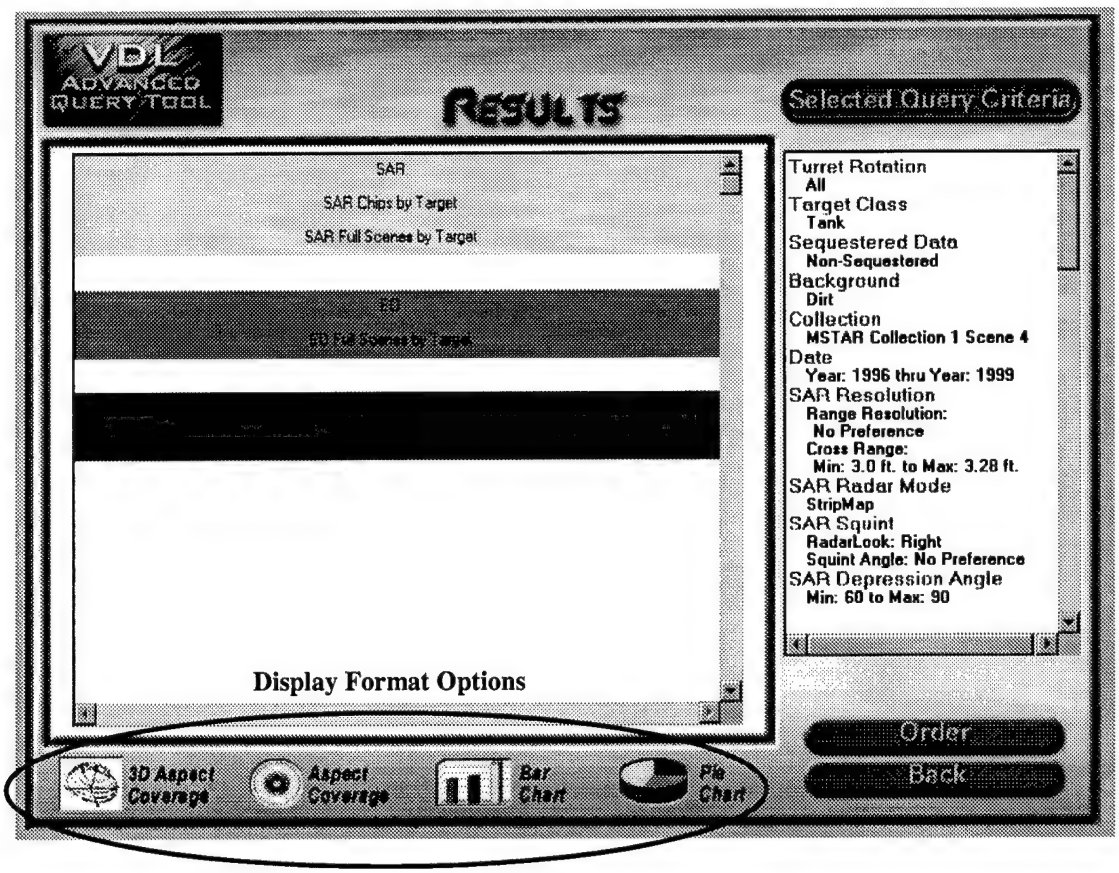

Figure 4: Advanced Query Tool 1.0 Results Screen Shot

conjunction with the AQT itself. The AQT Server Initial Configuration Program (shown in Figure 6) can be used to construct and publish a host default profile that describes the portion of the host's data holdings that are publicly available. This profile is created by configuring the database connection parameters and then selecting the fields within the metadata database(s) tables that are to be made available for queries by the end-users.

The resulting profile is then stored on a host server that is to execute the

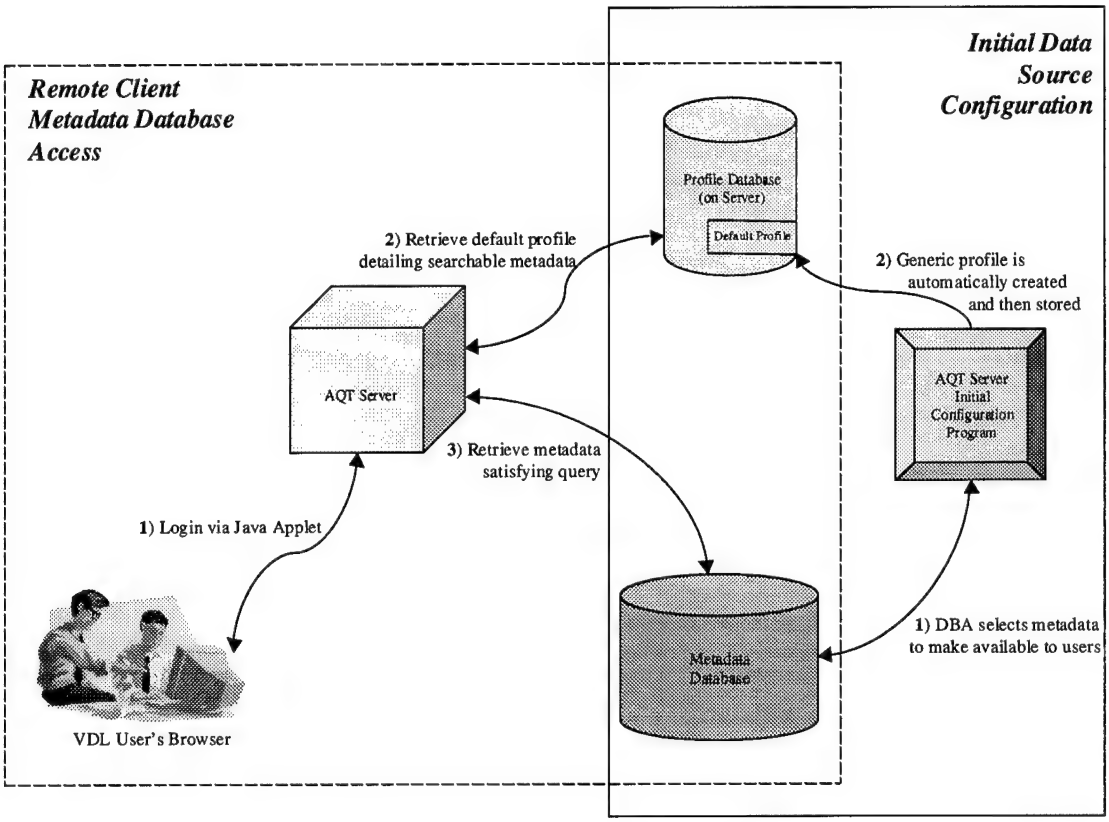

Figure 5: Advanced Query Tool 2.0 Functional Diagram

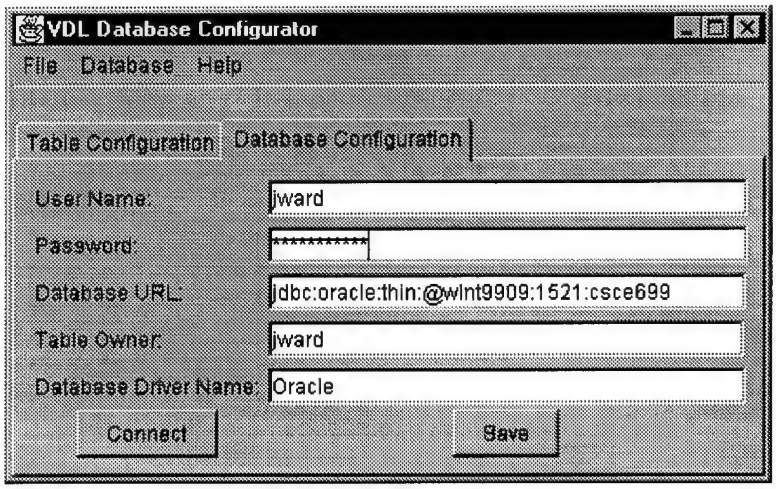

Figure 6: Advanced Query Tool 2.0 Server Initial Configuration Program

AQTServer process. This process will then serve on-request the default profile to client applets. Upon receiving the profile, a client's applet will look like Figure 7.

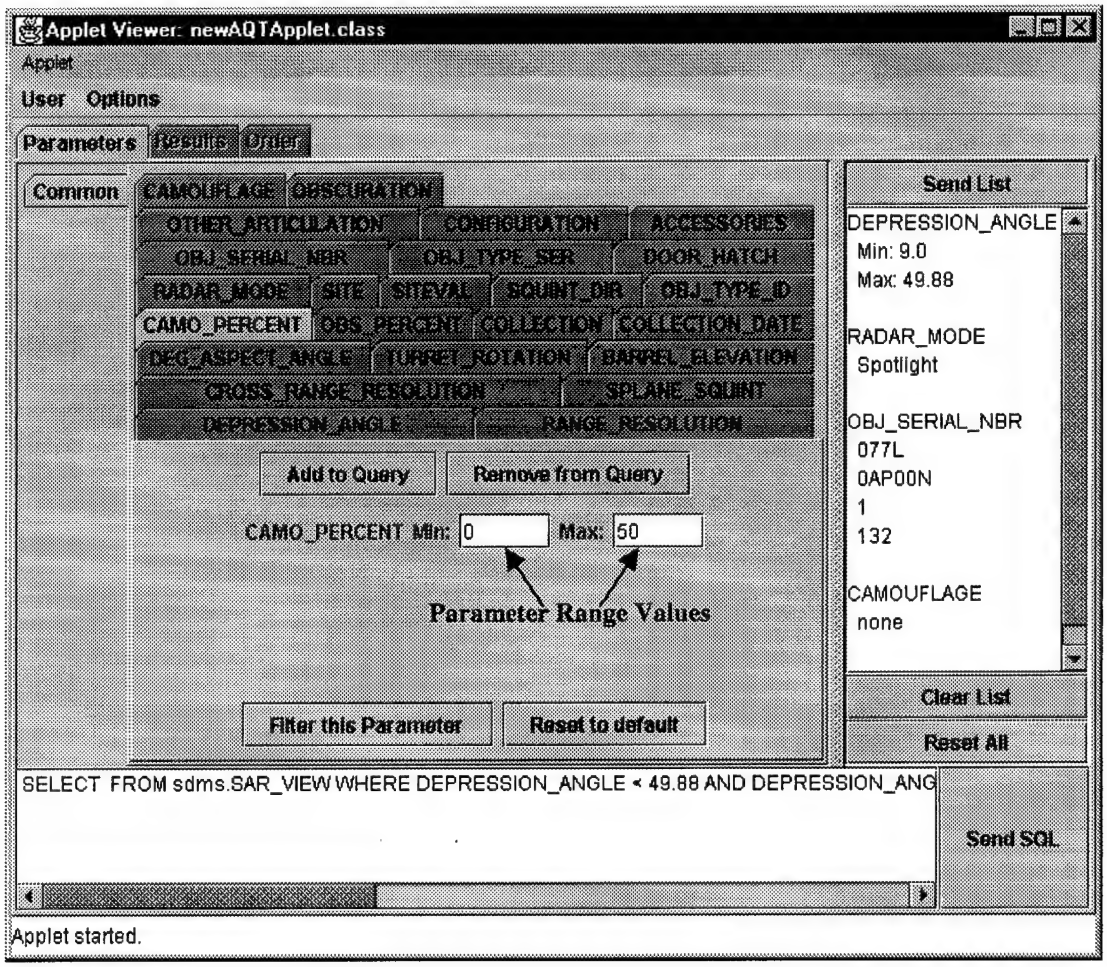

Figure 7: Advanced Query Tool 2.0 Client Applet

Query criteria can then specified by selecting the appropriate tabs, configuring the parameter range values, and selecting the "Add to Query" button. Once a user has finished entering query criteria, the user then selects the

"Send List" button. This causes the query to be sent to the AQTServer process in Figure 5. The server then retrieves the appropriate metadata from the database(s) and sends the results back to the client applet.

However, this is where the usability of the interface has unfulfilled potential. Once receiving the metadata result set satisfying the user's query, the user is unable to save (for later recovery) any of the information entered in constructing the initial query. Therefore, for complex or highly specific parameter selections, this requires complete reconstruction of the query. This issue will be discussed further in Chapter 3.

2.4 Usability Testing

Usability of an interface is often thought about in terms of "user friendliness." Specifically, usability testing is performed to determine the level of effort required by the user to learn the interface and perform tasks, as well as discover how error-prone the user is while using the interface. However, many aspects of usability are subjective since there are only limited objective standards by which to quantify an interface's usability. Previous work at the Air Force Institute of Technology (AFIT) performed by Captain Phillip Stratton [STRAT99] described four possible methods of analyzing usability that were introduced in papers by Matthias Rauterberg [RAUTE92][RAUTE95][RAUTE97] in an attempt

to make usability metrics less subjective. These methods are formal usability testing, user-oriented usability testing, product-oriented usability testing, and interaction-oriented usability testing. This research focused on using product-oriented usability analysis, and hence, only this type of analysis will be discussed here. For a summarizing discussion of other types of metrics-based analysis, see Captain Phillip Stratton's thesis [STRAT99].

2.4.1 Interaction Spaces

According to Rauterberg, product-oriented usability analysis involves quantitatively measuring the ergonomic attributes of an interface [RAUTE97]. Particularly, the ergonomic attributes of an interface include:

- Location and functionality of icons
- Display of information supporting the user's goals
- Degree of change due to context changes within the interface

By analyzing these individual aspects of the interface, unbiased information is obtained based upon a functionally-oriented method of evaluation.

In this method, all interfaces are composed of two different classes of components: objects and functions. Each of these components are either perceptible or hidden. If a component is perceptible, it can be manipulated within the interface. Otherwise it is hidden and cannot be manipulated. A

component has an additional boolean attribute of being either a dialog or an application. A dialog component allows the user to change the context of the current dialog in some way. An application component provides the user the ability to interact with the associated application, that is to modify through the interface the actual data being manipulated.

An example of a perceptible dialog object would be a button to resize the window or a visible menu item on a menu bar. A hidden dialog object would be an element of a drop-down box from the menu bar that is not currently visible. Perceptible functions are associated with the corresponding perceptible objects that allow the user to manipulate them. These are called perceptible function interaction points. One example of a perceptible function interaction point is at a "Save" button which changes the data in the underlying application. The user cannot actually manipulate the function but instead uses a perceptible application object (the "Save" button) to interface with the "Save" function. Each of these sets of objects and functions is broken out in Figure 8. Another representation of these sets is shown in Figures 9 and 10, described below.

2.4.1.1 Object Space

Objects in a user interface can be classified into the four categories shown in Figure 9. A perceptible object can be a perceptible dialog object (PDO) or a

perceptible application object (PAO), while a hidden object can be a hidden dialog object (HDO) or a hidden application object.

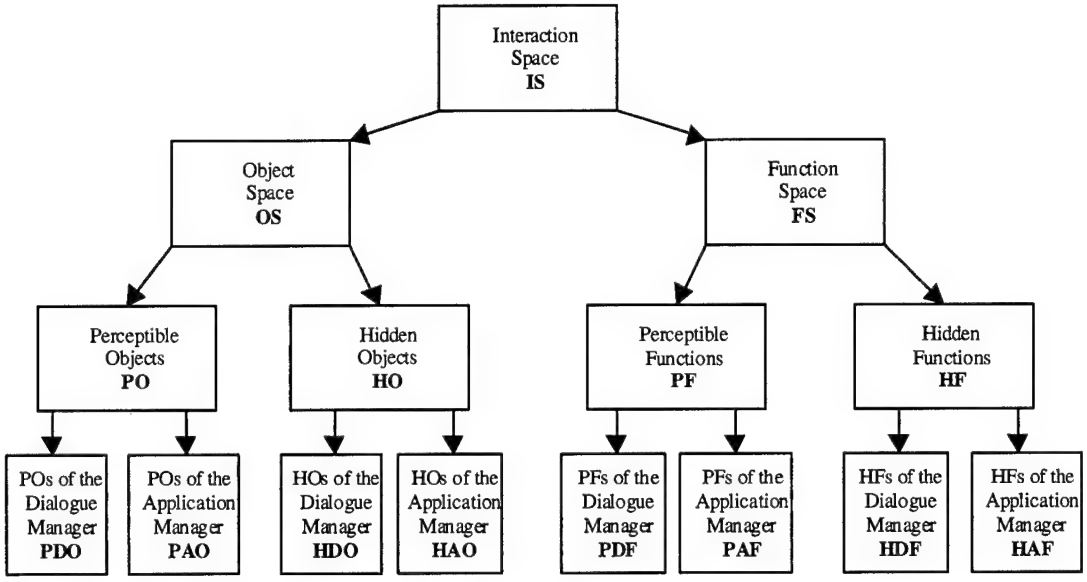

Figure 8: Graphical Representation of Interaction Space

Interface Objects

	Perceptible	**Hidden**
Dialog	Perceptible Dialog Objects	Hidden Dialog Objects
Application	Perceptible Application Objects	Hidden Application Objects

Figure 9: Interface Object Space

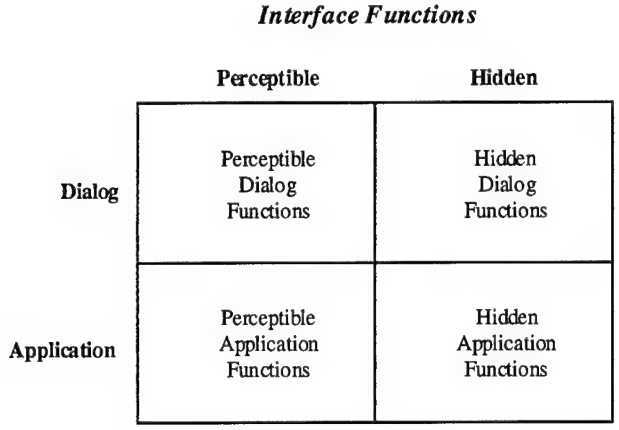

Figure 10: Interface Function Space

2.4.1.2 Function Space

Functions in a user interface are similar to objects in that they are restricted to the four classifications shown in Figure 10. However, functions are identified by the visible objects (or interaction points) with which they are associated in the interface. Therefore we have four types of functions: hidden dialog function interaction points (HDFIP), perceptible dialog function interaction points (PDFIP), hidden application function interaction points (HAFIP), and perceptible application function interaction points (PAFIP).

2.4.1.3 Dialog Context

Having defined object and function space, we can now define the interaction space as simply the cross product of object space and function space.

This provides a complete set of all the object and function pairs from which to select for each dialog context, where a dialog context is an element of the interaction space. Note that each user interface may have more than one dialog context. Therefore Equations 1 and 2 follow (where **IS** = Interaction Space and **DC** = Dialog Context).

IS = Object_Space X Function_Space (1)

DC ∈ IS (2)

2.4.2 Usability Metrics

Four metrics have been developed by Rauterberg to define usability: functional feedback, interactive directness, application flexibility, and dialog flexibility [RAUTE95]. Descriptions of each of these metrics follow.

2.4.2.1 Functional Feedback

Functional Feedback (**FB**), or the feedback to the user from the interface, is the sum of the ratios of the cardinality of perceptible functions to the cardinality of hidden functions of all the dialog contexts in the interface. In Equation 3, **D** is the number of dialog contexts, $\#PF_d$ is the cardinality of perceptible functions in a particular dialog context d and $\#HF_d$ is the cardinality of hidden functions in a

particular dialog context *d*. The higher the functional feedback of an interface, the greater the percentage of functions the user perceives.

$$FB = 100\% * \frac{1}{D}\sum_{d=1}^{D}\frac{\#PF_d}{\#HF_d} \qquad (3)$$

2.4.2.2 Interactive Directness

Interactive Directness (**ID**), or the depth that must be achieved before actually manipulating application data in the interface, is the sum of the lengths of all possible paths through the dialog contexts to a desired hidden function interaction point. In Equation 4, PATH is the set of all possible sequences of interactive operations from the top-level dialog context to dialog contexts within the desired hidden function interaction points. $PATH_p$ is the length of a path associated with p, **P** is the number of all those possible paths and *len* is the length of the particular sequence p. A higher value of interactive directness represents quicker access to information for the user. A command-line interface has an ID-value approaching 100%.

$$ID = 100\% * 1/\left\{\frac{1}{P}\sum_{p=1}^{P} len(PATH_p)\right\} \qquad (4)$$

2.4.2.3 Application Flexibility

Application Flexibility (**DFA**), or the flexibility of the interface with

respect to the application it supports, is the sum of the cardinality of the application function interaction points for all the dialog contexts in the interface. In Equation 5, **D** is the number of dialog contexts and $\#AFIP_d$ is the cardinality of application function interaction points for a particular dialog context d. An application function interaction point is a location on the interface (e.g. an icon) that is linked through a function to the application. For example, the X button often found on the top right corner of Microsoft Windows™ software has an application function interaction point connected with the close function of the application. A resulting value below 15 for this metric is not considered statistically significant [RAUTE95].

$$\mathrm{DFA} = \frac{1}{\mathrm{D}} \sum_{d=1}^{D} (\#AFIP_d) \qquad (5)$$

2.4.2.4 Dialog Flexibility

Dialog Flexibility (**DFD**), or the flexibility of the interface with respect to the dialog with the user, is the sum of the cardinality of the dialog function interaction points for all the dialog contexts in the interface. In Equation 6, **D** is the number of dialog contexts and $\#DFIP_d$ is the cardinality of dialog function interaction points for a particular dialog context d. A dialog function interaction

point is a location on the interface (e.g. icon or menu item) that is linked through a function to a change in the dialog context. A resulting value below 15 for this metric is not considered statistically significant [RAUTE95].

$$\mathrm{DFD} = \frac{1}{\mathrm{D}} \sum_{d=1}^{D} (\#DFIP_d) \qquad (6)$$

The main disadvantage of the product-oriented usability analysis method is that the user has no real input into how the values are determined (i.e. the user's feelings are not taken into account when determining the usability). This may be valuable since an unbiased view of the interface should provide a more accurate evaluation of the interface. However, the end-user will be the one using the interface, if he is not happy the product is not useful.

2.5 Summary

This chapter provided an outline of the query issues of digital libraries as well as user profiling technology. An overview of the architecture of the AFRL/SN Virtual Distributed Laboratory was also provided and product-oriented usability analysis metrics for the user interface were introduced. Chapter 3 establishes a methodology for inserting user profiling into the existing vision for virtual distributed library access, as well as describing how product-

oriented usability metrics can be used to objectively analyze the AQT 1.0 and a profiling-augmented AQT 2.0.

3 Methodology

Chapters 1 and 2 discuss query interfaces to distributed libraries and the current interface of the AFRL/SN Virtual Distributed Laboratory (VDL). In this chapter, we introduce a methodology for enhancing a virtual distributed library user interface through the use of profiles and for gauging interface usability via a metrics suite. While this research complements general user interface improvements, its primary focus is the successful integration of user profiling. The objective of this methodology is to help construct a robust query interface for digital libraries that will satisfy the requirements of Section 2.1 while quantitatively improving the user interface of AFRL/SN's VDL through a usability analysis method introduced in Section 2.4. This chapter outlines steps that can be used to reproduce this research on other virtual distributed libraries and query interfaces. Section 3.1 describes how to integrate user profiling into an existing Java™ applet-based virtual distributed library, such as the one detailed in Section 2.3 (AFRL/SN's VDL). Section 3.2 outlines the steps for performing a product-oriented usability analysis for an interface using the techniques discussed in Section 2.4. In Section 3.3, we summarize and close the chapter.

3.1 User Profiling Insertion

The advantages of user profiling and developing a relationship with the

end user are discussed in Section 2.2. In summary, every interactive session with a user provides a chance to augment the services offered by a virtual distributed library in subsequent sessions. This section provides a description of the hardware and software configuration of the research in Section 3.1.1, details user profiling implementation (including server-side and client-side caching of profiles) in Section 3.1.2, and describes enhancements user profiling may provide to central hosts in Section 3.1.3.

3.1.1 Hardware and Software Configuration

The source code was written in Java™ 1.2 and provided in its original forms (*Advanced Query Tool* (AQT) 1.0 and 2.0) by the AFRL/SN VDL. The code was subsequently compiled in the Microsoft Windows™ operating system using the Java Development Kit™ 1.2.2. For details on gaining access to the source code, see Appendix A.

3.1.2 Storage Location of User Profiles

Two different methods are to be considered for the storage location of the user profiles: client-side storage and server-side storage. Each of these locations has advantages and drawbacks that are discussed in their corresponding subsections.

3.1.2.1 Client-Side Storage of Profiles

The most straightforward client-side solution would be implemented through the use of "cookies"—modifiable data files stored on and visible to the client, yet accessible by the server via client-side applets for maintaining state and session information. The greatest benefit from this approach is the access granted to individual users, permitting them to change their own profiles offline (either though direct text manipulation or a user-friendly application possibly downloaded from the server). Furthermore, this approach saves disk space on the server, as well as bandwidth between the client and server that would have been used accomplishing profile management.

With this client-side access, though, comes additional security concerns. Without additional security implementation (and overhead) on the server-side, creative users with a malicious intent may be able to spoof servers into revealing portions of the metadata database(s) that the administrator had not intended to permit access.

3.1.2.2 Server-Side Storage of Profiles

This security issue can be remedied by selecting server-side storage and maintenance of user profiles. Using this method, profiles are stored on and retrieved from the server directly, allowing no outside access other than

manipulation through the AQTServer application, itself. This security requirement is critical to the successful accomplishment of the AFRL/SN mission for the VDL, thus, this methodology incorporates a server-side storage of profiles. We do not see the potential savings in bandwidth and storage space as strong enough arguments to risk the security implications of a client-side solution in the foreseeable future, as long as a user-friendly interface is implemented to allow one to modify his user profile as managed by the server.

This decision to implement server-side caching of profiles led to a new AQT design shown in Figure 11.

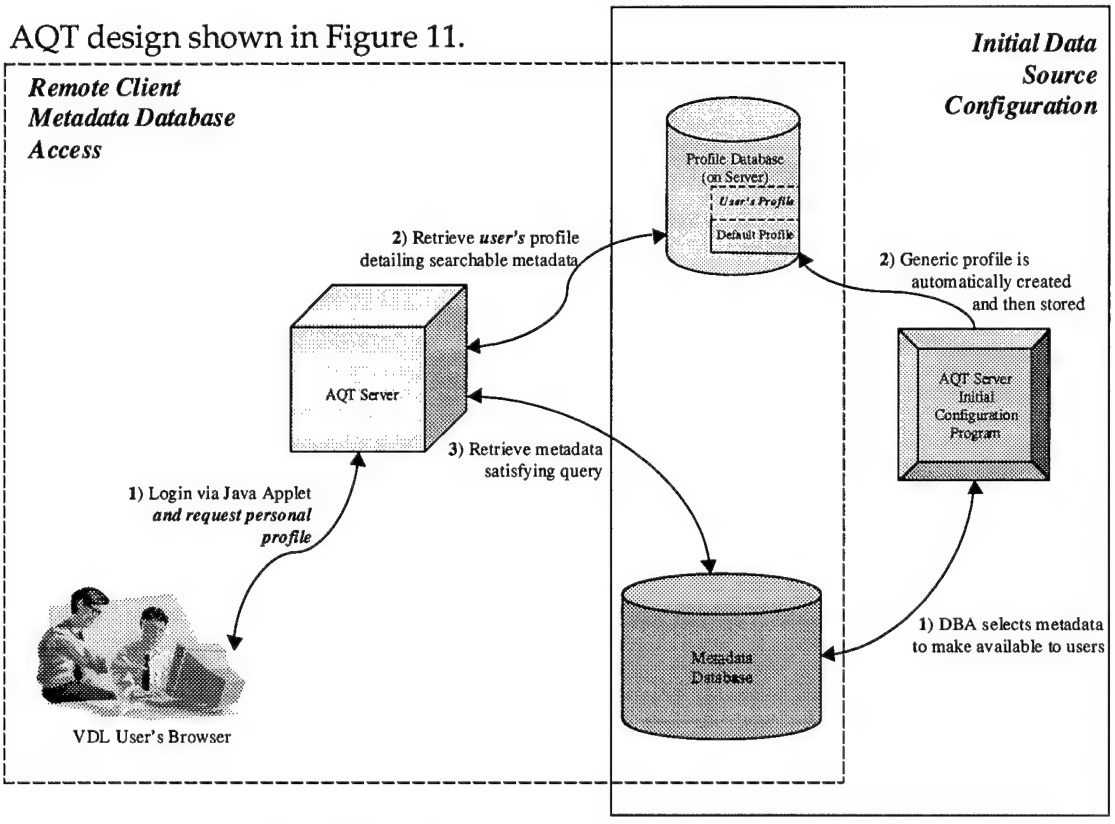

Figure 11: User Profile-Enhanced AQT 2.0 Functional Diagram

Notice how a user's profile is now stored in a profile database on the server-side and delivered to the client upon request. In our experiment, the profile database consisted of flat files stored in a single directory, but a more extensive digital library with multiple and concurrent users would require incorporation of a database management system for the maintenance of profiles.

3.1.3 Enhancing Central VDL Capability

To complement the augmented query interface, user profiling provides the capability of enhancing the functionality of the central host of virtual distributed libraries. Figure 12 depicts the information flow after implementation.

This flow of information to the central host provides additional client-related transactional data which can be leveraged by the VDL host. By having each remote site send all of its user information and metadata of query result sets to a central virtual distributed library server, one would now provide the central host with unprecedented critical management functionality. This functionality would include a reporting capability across the DoD. For example, consider reports (by library, by customer, by MAJCOM, by contractor, and of overall datasets) highlighting numbers of queries satisfied, image sets most heavily sought, and most queried parameters. Numerous significant underlying trends

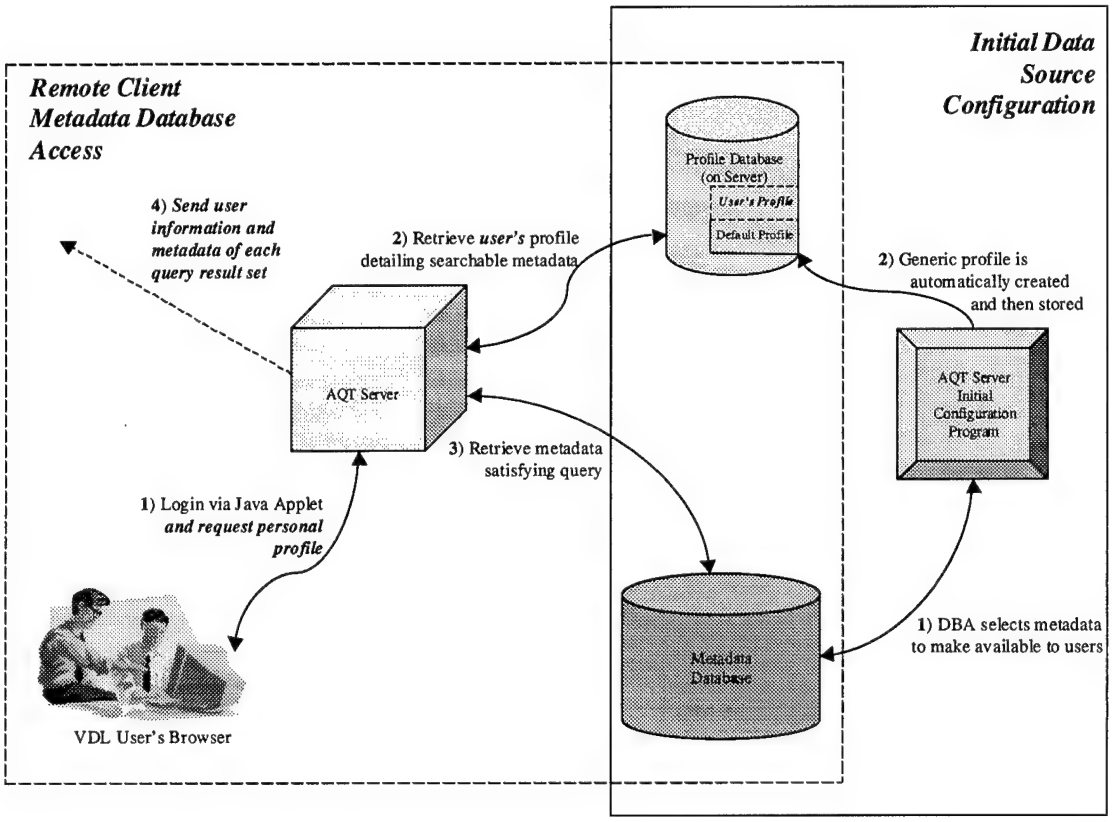

Figure 12: User Profile-Enhanced Distributed Library Functional Diagram

may be identified and utilized to optimize future query capability at the remote locations.

Furthermore, central recording and interpretation of this information would also enable high-end management functionality, such as bandwidth management and ultimately "pay-per-byte" infrastructures. That is, the central host could monitor remote client usage statistics and "bill" customers accordingly. This potential "revenue" flow (in dollars or other entitlements)

could be used to improve the quality of data holdings, upgrade links to the remote sites, or perhaps further virtual distributed library software development.

3.2 Interface Analysis

With the development of an augmented query interface for the digital library, an analysis can now be conducted to verify the increased usability of the interface using product-oriented usability analysis methods as discussed in Section 2.4. There are four steps to implementing the metrics described in Equations 3 – 6 from Chapter 2:

1) Develop a dialog tree showing the interaction between dialogs within the interface.

2) Analyze the dialog tree in order to determine the number of dialog contexts, number of paths through the interface, length of each path, and cardinality of dialog function interaction points.

3) Analyze the source code to help determine the cardinality of hidden and perceptible functions as well as the cardinality of application function interaction points.

4) Calculate values for the metrics using Equations 3 – 6.

The following subsections detail the procedures for each step.

3.2.1 *Dialog Tree Development*

A dialog tree is an n-ary tree in which each node represents a dialog of the interface. An example is depicted in Figure 13.

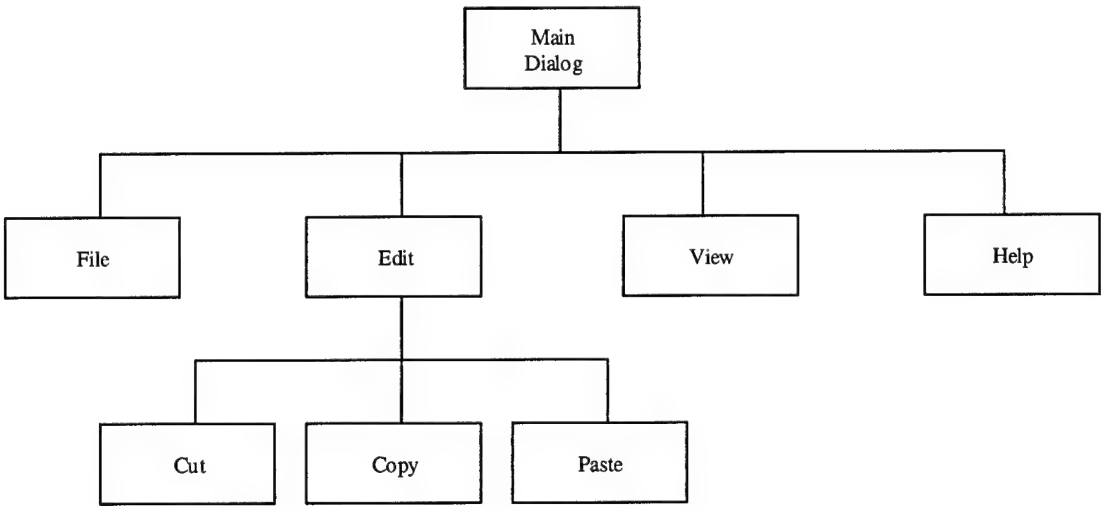

Figure 13: Diagram of Dialog Tree

Each connection in the tree represents a dialog function interaction point that connects a parent dialog context with a child dialog context. Using Figure 13 as an example, the main dialog has four dialog function interaction points (or buttons): File, Edit, View, and Help. The File button has no child dialog contexts linked with it. The button for Edit would open a new dialog context from which the user can choose Cut, Copy, or Paste. The View and Help buttons are also not related to child dialog contexts.

3.2.2 *Deriving Values From the Dialog Tree*

In order to calculate specific values for the metrics equations, we use the dialog tree. The number of paths in the interface, **P**, is simply the number of leaf nodes in the dialog tree. To clarify the definition of a path, consider that a path in the interface is a sequence of events that must happen for a user to actually interact with the application, not just the dialog contexts of the interface. Also, each leaf node in the dialog tree corresponds to the final dialog where the interface interacts with the application.

The total number of dialogs in the interface, **D**, is also easy to determine. Since each node in the dialog tree represents a dialog in the interface, the total number of nodes in the dialog tree is equivalent to the number of dialogs in the interface.

The length of each path, $len(PATH_p)$, is also simple to ascertain. The length of each path is the depth of each node in the dialog tree, because each path terminates at a leaf node in the tree.

The cardinality of the dialog function interaction points, $\#DFIP$, associated to each dialog is the final value that can be established directly from the dialog tree. This value is determined by counting the number of children per node of the dialog tree. Here, each connection to a child node represents a dialog

function interaction point of the parent.

3.2.3 *Deriving Other Interface Values*

Having derived these values from the dialog tree, three more values remain to be determined to enable the calculation of the metrics outlined in Section 2.4. These three values are not as simple to establish as the ones discussed previously.

The first value is the cardinality of perceptible functions, $\#PF$, in the interface. This value is found by determining the total number of functions the user can manipulate in each dialog. Mathematically, this value is just the cardinality of the union of perceptible dialog and application function interaction points for a given dialog context. Thus, a careful inspection of the source code, if available, and a close examination of the operation of the interface should reveal the true number of perceptible functions.

The cardinality of hidden functions, $\#HF$, is often more difficult to assess. Mathematically, this value is the cardinality of the union of hidden dialog and application function interaction points for a given dialog context. Due to these functions being hidden, this value cannot be determined by looking just at the interface. However, a thorough examination of the interface coupled with a close

inspection of the source code should reveal this value.

Lastly, the cardinality of application function interaction points, $\#AFIP$, in the interface is the count of interaction points in the interface that perform actual functions within the application. Normally, application function interaction points only occur in the leaf nodes of the established dialog tree, such as "Cut" in Figure 12. Nonetheless, occasionally there are hidden application function interaction points of which the user is unaware. These hidden application function interaction points perform operations in the background and can only be determined with an exhaustive examination of the source code of the interface.

3.2.4 Functional Feedback

Now that we have explained how to measure the values involved in the calculation of these metrics, we will discuss the methods and reasoning for improving *Functional Feedback*, as defined in Equation 3 (repeated below), of an interface.

$$FB = 100\% * \frac{1}{D}\sum_{d=1}^{D}\frac{\#PF_d}{\#HF_d} \qquad (3)$$

By increasing the functional feedback metric, we will amplify the amount

of response (i.e. feedback) that the interface provides to the user, increasing the user's confidence in the functionality of the interface. Since there are three variables that determine this metric, there are three possibilities to improve the usability as measured by this single value.

One option is to manipulate the number of dialogs, **D**, in the interface. However, this may not be a viable method for improving this metric, since decreasing the number of dialogs requires increasing the number of interaction points per dialog, $\#PF_d$ and $\#HF_d$, in order to maintain the same functionality. An increase in perceptible functions, $\#PF_d$, will result in a "busier" interface that will be less usable, while an increase in hidden functions, $\#HF_d$, will undermine a user's understanding of the interface's functionality.

Another option is to increase the number of perceptible functions, $\#PF_d$. However, too many perceptible functions will only serve to clutter dialogs and confuse the user, thus reducing the overall usability of the interface (although the metrics would reflect better numbers).

The final option for improving functional feedback is to decrease the number of hidden functions, $\#HF_d$, within the interface. By design, hidden functions conceal information from the user when these functions are executed automatically from within previously-selected perceptible functions. For

example, the automatic saving of all documents in a word processor regardless of size may seriously hinder the interface's usability due to its sudden unresponsiveness. By providing feedback to the user for as many functions as possible, the user is more knowledgeable of events within the interface, improving both the functional feedback metric and overall measured usability for the end user.

3.2.5 Interactive Directness

Interactive Directness, is the metric that the inclusion of user profiling should most enhance, since it is a measure of how many dialogs must be traversed before the user is able to undertake the desired function. The interactive directness metric, as defined in Equation 4 (and re-shown below), may be improved using the following three approaches.

$$ID = 100\% * 1/\left\{\frac{1}{P}\sum_{p=1}^{P} len(PATH_p)\right\} \qquad (4)$$

One approach is to decrease the number of paths, **P**, in the interface. This may, however, result in confusing the user, since there may now be totally unrelated functions within a particular dialog context. The second method is to decrease the length of each path, $len(PATH_p)$, within the interface. This may have the same effect as reducing the number of paths since the user will be confronted with more function interaction points per dialog context, if functionality remains

unchanged.

A third method, that should reliably improve this metric, is to utilize user profiling to decrease the number of visible paths based on the user's previous preferences. In this manner, a user can access the exact dialog context required more rapidly and reliably, but retaining the same functionality by permitting the user to change and subsequently store his visible path preferences.

3.2.6 Application and Dialog Flexibility

Improvements in these last two metrics, *Application Flexibility and Dialog Flexibility* (in Equations 5 and 6, re-shown below), are difficult to achieve.

$$\mathrm{DFA} = \frac{1}{\mathrm{D}} \sum_{d=1}^{D} (\#AFIP_D) \qquad (5)$$

$$\mathrm{DFD} = \frac{1}{\mathrm{D}} \sum_{d=1}^{D} (\#DFIP_D) \qquad (6)$$

By simply examining these equations, it appears that by increasing the average number of dialog or application function interaction points per dialog context, one could increase these metrics respectively. Nevertheless, as stated previously, maintaining functionality while increasing the number of functions per dialog will bombard the user with unrelated functions within a smaller

number of dialogs, leading to an overall less usable interface.

3.3 Summary

This chapter has introduced a methodology for augmenting a virtual distributed library such as AFRL/SN's VDL with server-side caching of user profiles. A analysis of client-side caching was also provided. Furthermore, central virtual distributed library capabilities were expounded that permit the collection and utilization of detailed user information and queries to provide reporting functions and future system enhancements. In addition, this methodology has elaborated the decision points and ultimately distilled verifiable, quantitative improvements to the query interface of a digital library. These steps included the elaboration of four metrics designed to illustrate the usability of an interface. Care must be taken to ensure that any attempt to increase one of these metrics is implemented with groupings of similar functions to prevent unintentional usability inefficiencies. In the next chapter, the steps of this methodology are applied to the VDL's *Advanced Query Tool*, and the results are shown and subsequently analyzed.

4 Case Study Analysis and Results

Chapter 3 presents a methodology to enhance query interfaces to virtual distributed libraries and measure the quantitative improvements to the interface using product-oriented usability analysis. This chapter outlines an application of that methodology to a test case by presenting the implementation and subsequent analysis of a user-profile augmented version of the AFRL/SN's *Advanced Query Tool*. Section 4.1 discusses the implementation of user profiling in the AQT 2.0. Section 4.2 provides an analysis comparing the original interface of the AQT 1.0 to the new interface of the AQT 2.0 and the resulting interface of the AQT from the application of our methodology. Section 4.3 summarizes our results.

4.1 Augmented AQT 2.0 Interface

The methodology for improving a virtual distributed library query interface laid out in Section 3.1 was implemented utilizing user profiling within the existing interface of the AFRL/SN Virtual Distributed Laboratory (VDL) AQT 2.0. Section 4.1 is divided into two subsections. Section 4.1.1 illustrates the method by which users may now save the visible data entry fields and fields used to retrieve the results of queries. Section 4.1.2 describes how a modification to the existing design of the *Advanced Query Tool* would harness more of the

enormous potential offered by user profiling by allowing end users to save the actual parameters used in building detailed queries.

4.1.1 User Profiling Implementation in the AQT 2.0

The next series of figures demonstrates how end users of the AFRL/SN's VDL may save the visible data entry fields and fields used to retrieve the results of their queries. Figure 14 shows the drop-down menu option used to enter a dialog (Figure 15 shows the actual dialog) to set the visible data entry fields. Figure 16 shows the result of selecting only the first two fields (Collection and

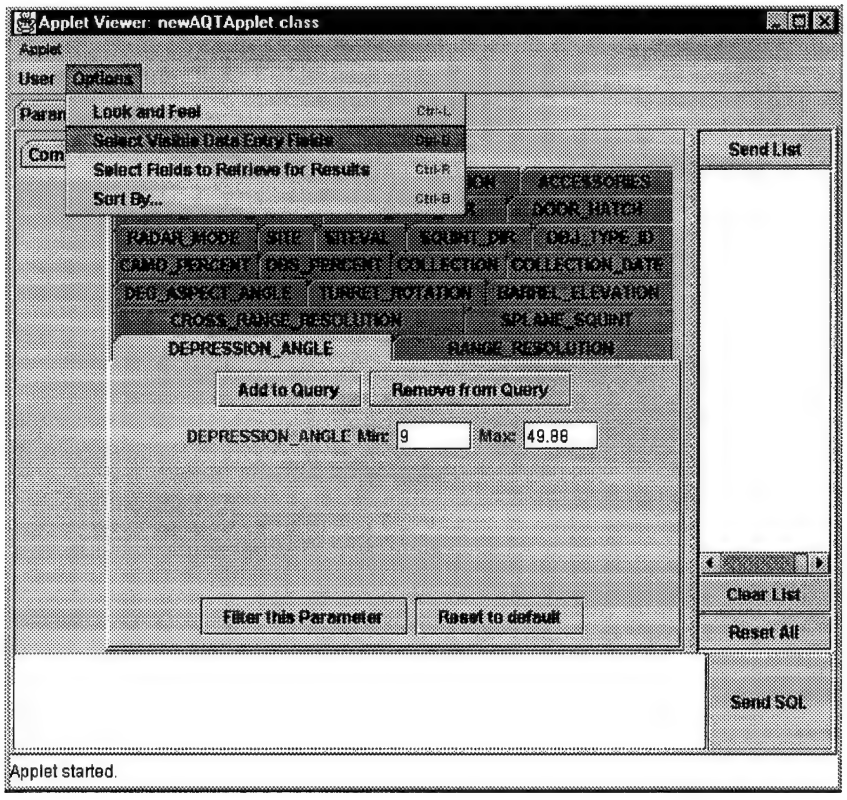

Figure 14: AQT 2.0 Option To Select Visible Data Entry Fields

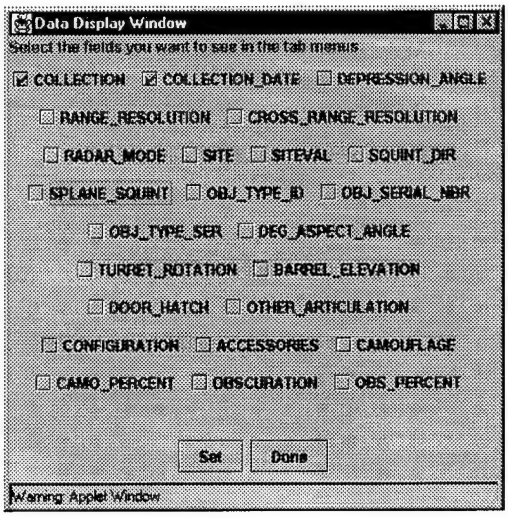

Figure 15: AQT 2.0 Dialog For Selecting Visible Data Fields

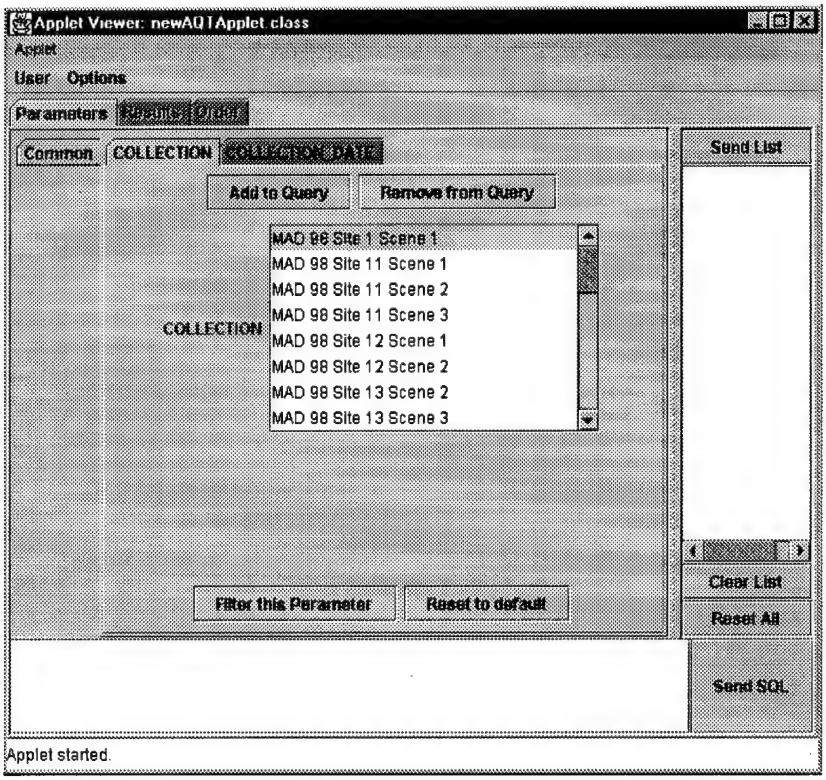

Figure 16: AQT 2.0, Post Visible Fields Selection

Collection_Date) to remain visible. Figure 17 shows the drop-down menu option used to enter a dialog (Figure 18 shows the actual dialog) to set the fields used to retrieve the results of the query (which only changes background variables not

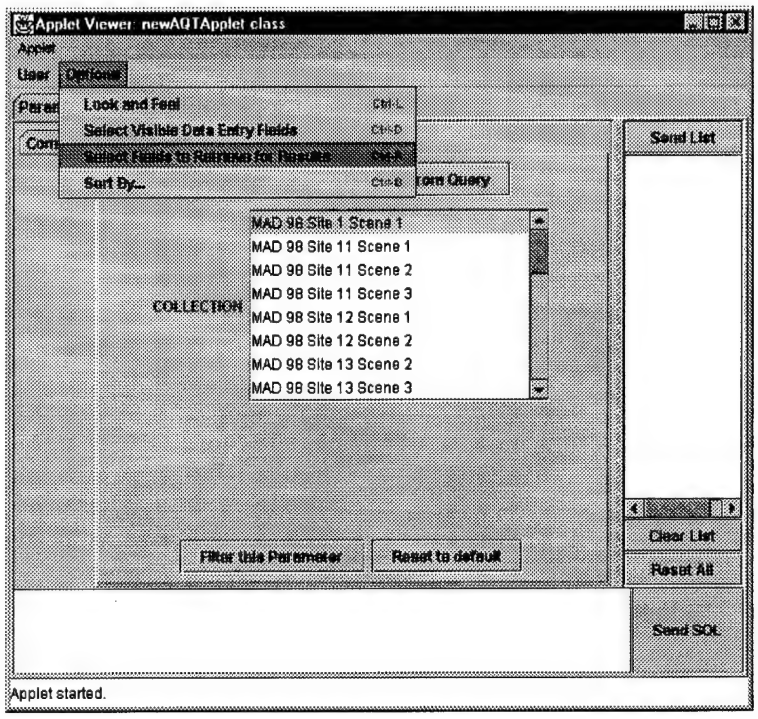

Figure 17: Option To Select Fields To Retrieve For Results

visible to the user). In this example, the user has chosen only to use the Collection and Collection_Date fields to specify parameters for the query. Before the implementation of this research, the user would be unable to save the visible data entry fields and the fields used to retrieve the results of the query. However, Figure 19 shows the option a user may now select in order to save his configuration as a profile to be used in subsequent sessions.

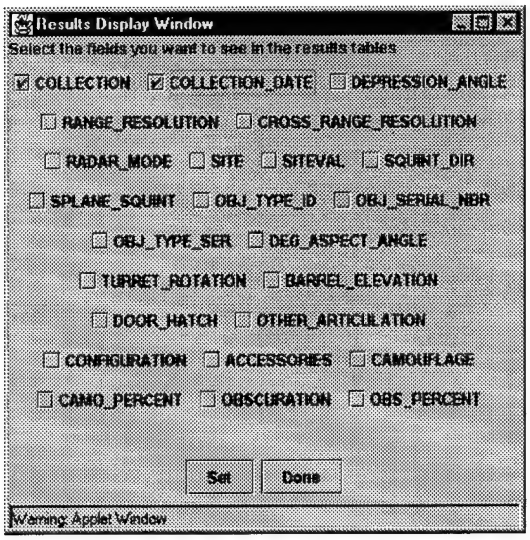

Figure 18: AQT 2.0 Dialog For Selecting Fields Used For Results

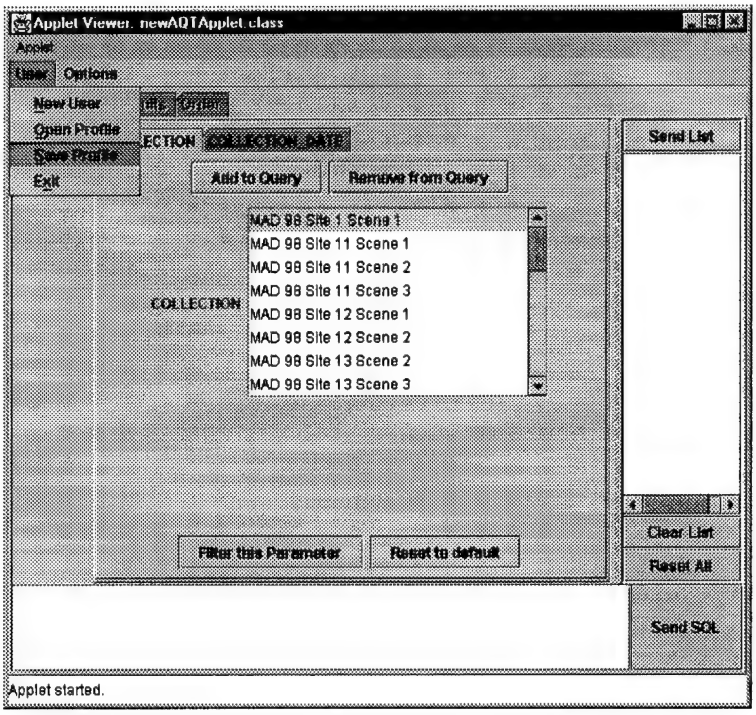

Figure 19: AQT 2.0 Save Profile Option

After completing the dialog (shown as Figure 20), the user will be able to bypass the tedious interactivity with the interface required to identify the fields to be used in satisfying the present and future detailed imagery and electrooptical signature data queries.

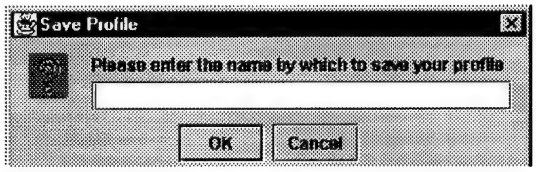

Figure 20: Save User Profile Dialog

The profiles are stored on the machine executing the AQTServer application. They are in the format of the original default profile created by the AQT Server Initial Configuration Program. As an example profile, the original default profile is shown in Appendix B. To explain what portions of the personal profile are now modified, consider this excerpt of the default profile:

```
COLLECTION.SAR_VIEW,COLLECTION,list,null,true,false,MAD 98 Site 1 Scene 1,MAD
98 Site 11 Scene 1,MAD 98 Site 11 Scene 2,MAD 98 Site 11 Scene 3,MAD 98 Site 12
Scene 1,MAD 98 Site 12 Scene 2,MAD 98 Site 13 Scene 2,MAD 98 Site 13 Scene
3,MAD 98 Site 6 Scene 1,MAD 98 Site 9 Scene 1,MSTAR Collection 1 Scene 1,MSTAR
Collection 1 Scene 2 Subscene 1,MSTAR Collection 1 Scene 2 Subscene 2,MSTAR
Collection 1 Scene 2 Subscene 3,MSTAR Collection 1 Scene 3,MSTAR Collection 1
Scene 4,MSTAR Collection 2 Scene 1 Subscene 1,MSTAR Collection 2 Scene 1
Subscene 2,MSTAR Collection 2 Scene 1 Subscene 3,MSTAR Collection 2 Scene 1
Subscene 4,MSTAR Collection 2 Scene 1 Subscene 5,MSTAR Collection 2 Scene
2,MSTAR Collection 2 Scene 3,MSTAR Collection 2 Scene 4,MSTAR Collection 3
Scene 1,MSTAR Collection 3 Scene 2,MSTAR Collection Squint
COLLECTION_DATE.SAR_VIEW,COLLECTION_DATE,list,null,true,false,19950825,19950831
,19950901,19950902,19950903,19950904,19950905,19961119,19961120,19961121,199611
22,19961123,19970320,19970321,19970508,19970509,19970510,19970511,19970513,1998
0427,19980504,19980506,19980508
```

These are the two entries for the Collection and Collection_Date fields

(whose titles are italicized in the excerpt). They hold all of the parameters and variables used by the AQT for configuring valid, permissible queries for the user. Notice the four boolean values in bold. The first boolean value in each entry represents whether the data field is visible to the user, while the second value reveals whether the user wants this field to be used when processing the query. In our example, these boolean values would be saved as "true,true" for the Collection and Collection_Date entries to reflect the current desire of the user to utilize these values in determining the result set.

Clients may now use the profiles they have previously stored on the server instead of the default profile by specifying one by name when starting the applet. Appendix A outlines the method to obtain the Java™ source code used to implement user profiling.

4.1.2 Fully Realizing User Profiling Potential

Although users may now save their detailed selections for visible entries and for fields used to satisfy queries, they are still required to re-accomplish, per query session, the specification of signature data parameters within the fields to be used. This becomes very significant when numerous data field entries must be edited. For example, consider the data field in this excerpt from Appendix B:

```
DEPRESSION_ANGLE.SAR_VIEW,DEPRESSION_ANGLE,minmax,null,true,false,9,49.88
```

The title of this field is Depression_Angle (and is italicized in this excerpt). Notice the two bolded values at the end. These two values will become the default minimum and maximum values for this item if inserted into a query (as shown in Figure 21).

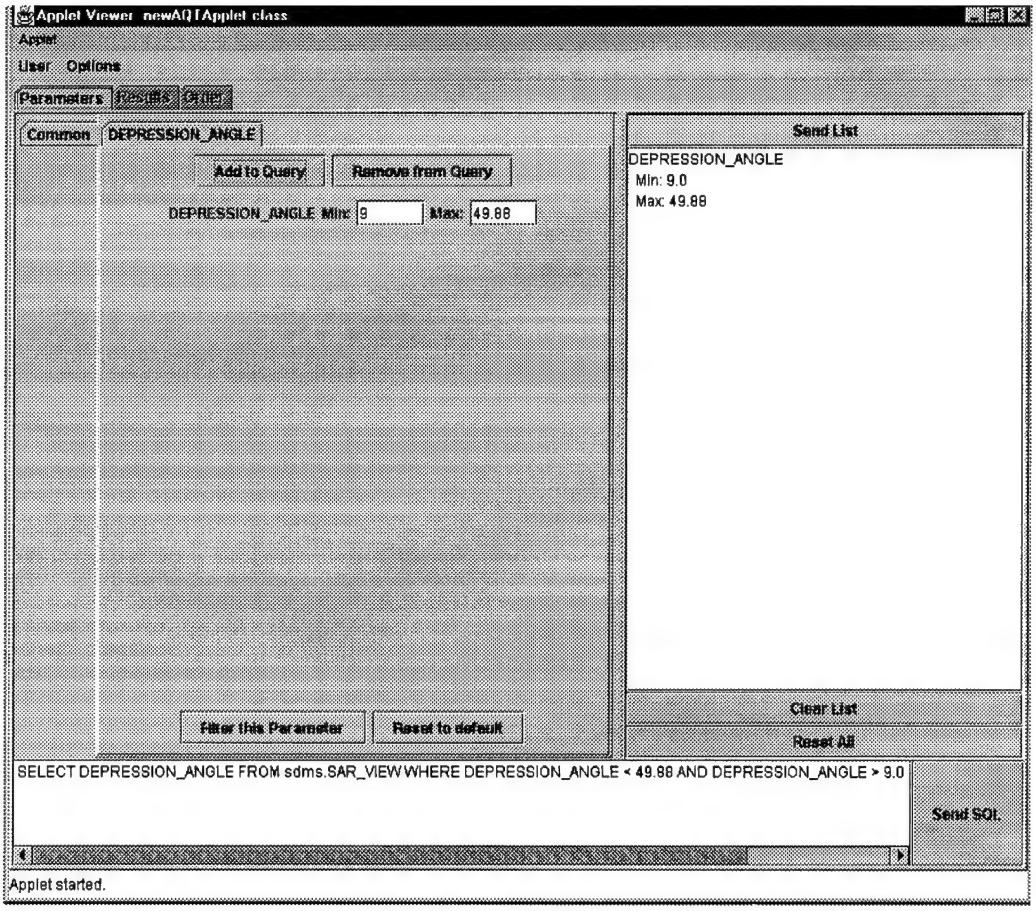

Figure 21: Example Using Default Parameters For Query

If the user wants to build queries with these two parameters in a narrower range, he may specify them by replacing the two values (9, 49.88) inside their corresponding data boxes with new values (e.g. 15, 20). However, these values

cannot be saved beyond the end of the session, due to the design of the AQT 2.0. Specifically, the format of the profiles and the data structures used throughout the client application preclude an easy implementation of this desired feature. Thus, we recommend a design change to the *Advanced Query Tool* that will accommodate the extrapolation and recording of these parameter range values in future development iterations. This change, if introduced, will slightly increase the size of profiles (by 2-4 bytes per data entry) and require a small addition of programming logic (by approximately 10 lines of code) into the processing of profiles section within the AQT.

4.2 Analysis of Advanced Query Tool Interfaces

Having completed the interface modification, the next step of the methodology is to quantitatively verify improvements to the interface by developing extensive dialog trees for the interfaces involved. Figures 22, 23, and 24 depict dialog trees for the AQTs 1.0, 2.0, and modified 2.0 respectively. We used the default profile (shown in Appendix B) to generate these trees and assumed the user was interested in using the Collection and Collection_Date as the parameters by which to specify the query. One should notice that the dialog tree for the augmented AQT 2.0 is much smaller, due to the use of a user profile specifying only the Collection and Collection_Date parameter fields as visible.

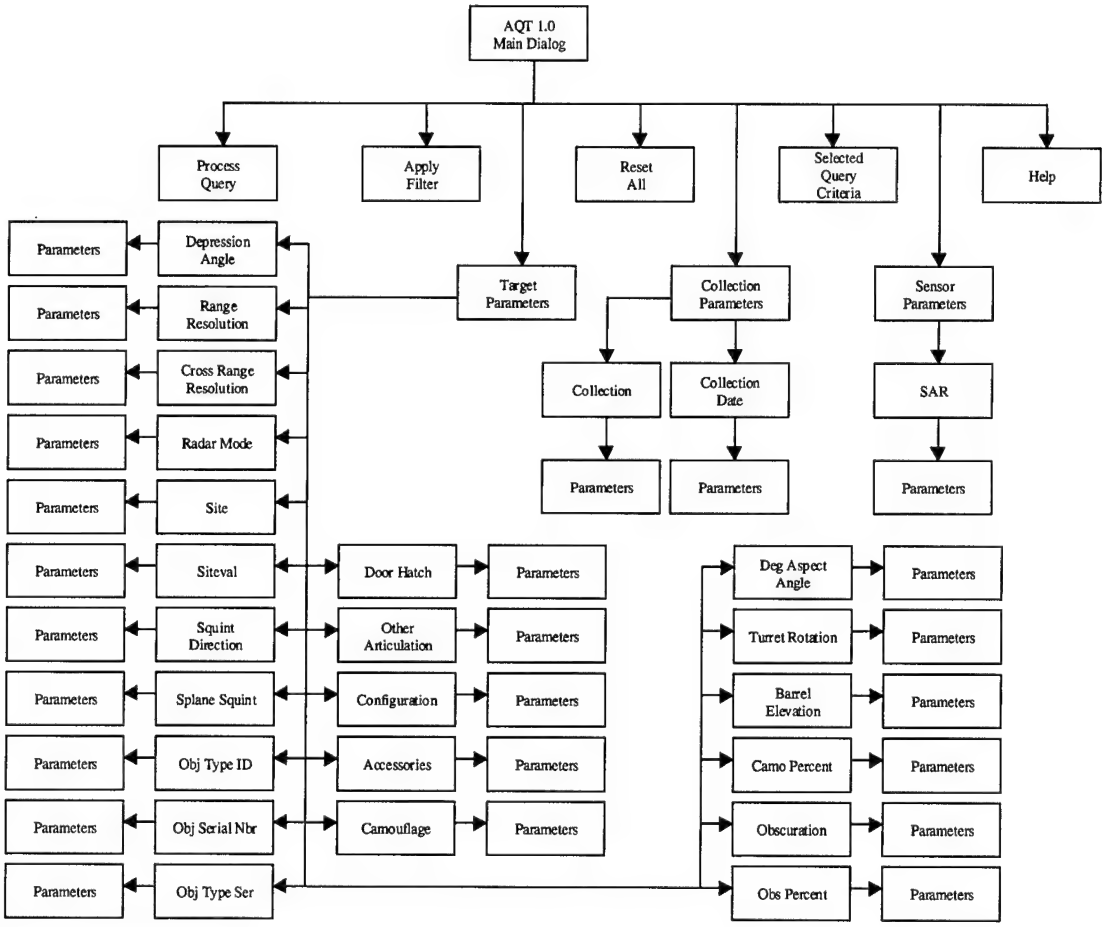

Figure 22: Dialog Tree for Advanced Query Tool 1.0

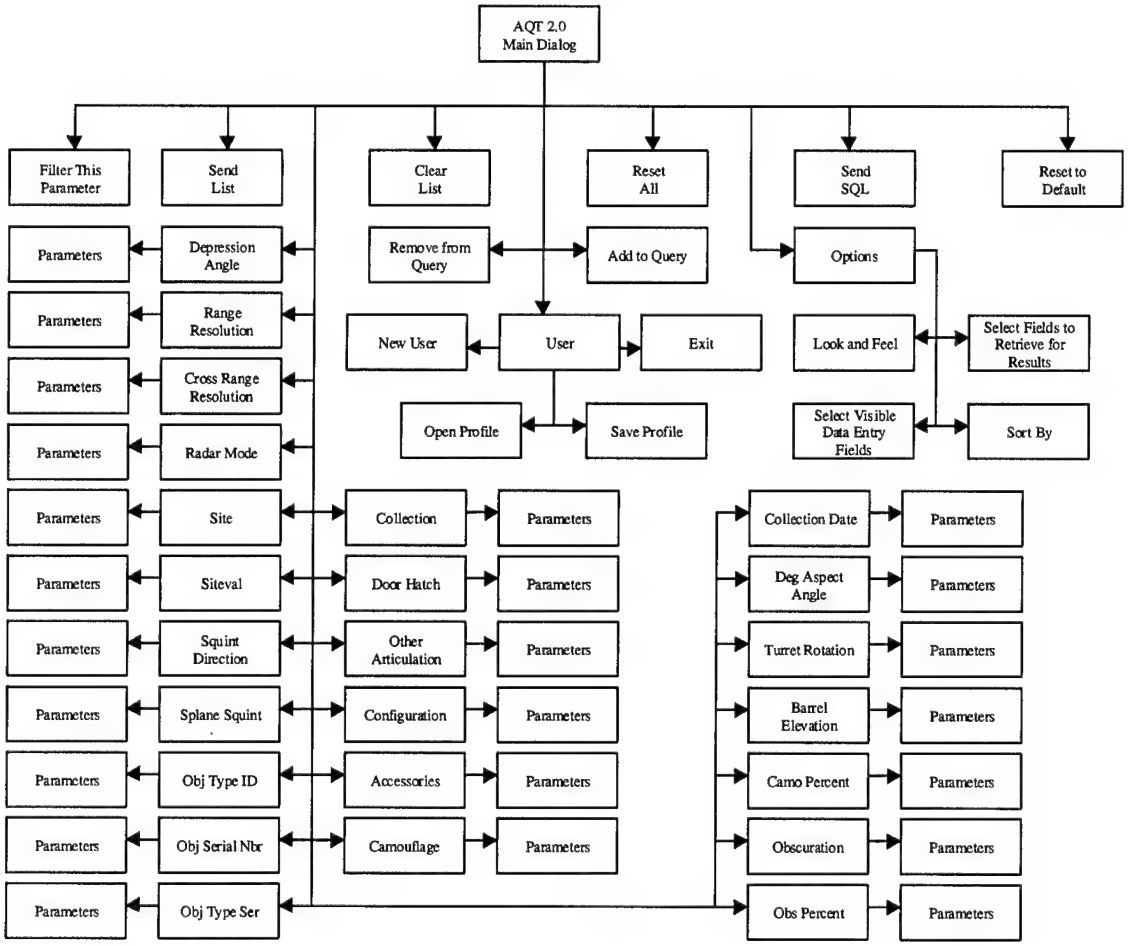

Figure 23: Dialog Tree for Advanced Query Tool 2.0

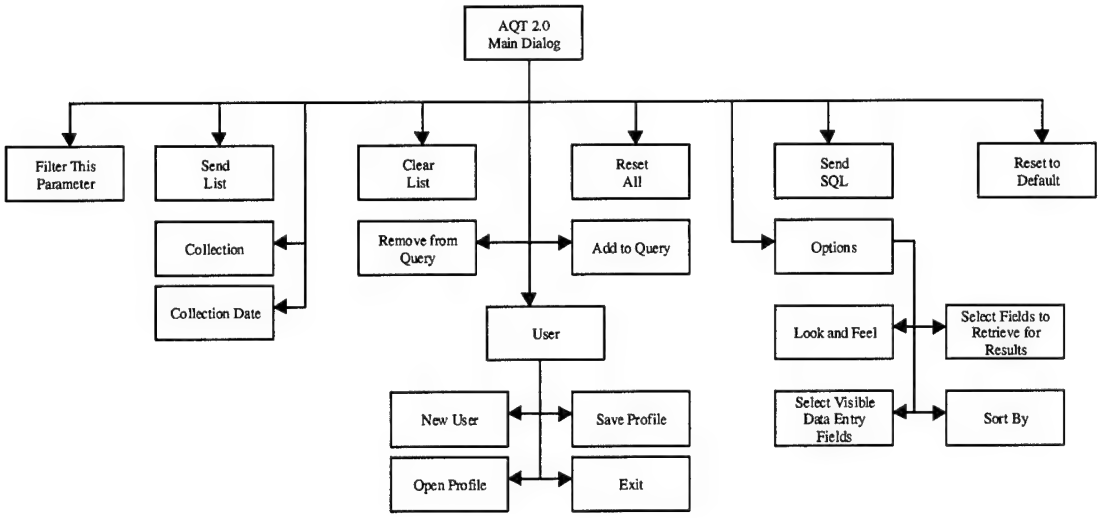

Figure 24: Dialog Tree for Augmented Advanced Query Tool 2.0

Using these dialog trees and the source code, the results for the analysis data of the three interfaces are compiled into Table 1. By applying the metrics equations (Equations 3-6) as discussed in the methodology in Section 3.2, Table 2 could be generated. As discussed previously, larger values for all of the metrics are preferable.

	D	P	*len(Path)*	*#PF*	*#HF*	*#DFIP*	*#AFIP*
AQT 1.0	59	30	110	195	52	133	114
AQT 2.0	67	40	112	166	43	98	111
Augmented AQT	21	18	44	50	45	28	67

Table 1: Product-Oriented Usability Analysis Data

	FB	ID	DFA	DFD
AQT 1.0	6.36%	27.27%	1.93	2.25
AQT 2.0	5.76%	35.71%	1.66	1.46
Augmented AQT	5.29%	40.91%	3.19	1.33

Table 2: Product-Oriented Usability Metrics Values

By examining the results of the application of these metrics, we can deduce how the design of the AQT affected the user interface. The values for *Functional Feedback* (**FB**) are in a very tight range, with the original AQT 1.0 having the highest value. This is mainly due to the fact that there are more methods to achieve the same functionality in the AQT 1.0. The AQT 2.0 **FB** value is slightly higher than the Augmented AQT value, because there are more parameters to be adjusted initially without the implementation of user profiling.

The most significant result of this portion of the research is the improvement in the *Interactive Directness* (**ID**) metric due to the inclusion of user

profiling in the Augmented AQT. The increase in **ID** is 5.20% from the AQT 2.0 to the Augmented version, while the increase from the original AQT 1.0 to the Augmented version is 13.64%. This 13.64% increase in **ID** represents an impressive 50.02% improvement over the **ID** for the original AQT 1.0 (27.27%). Moreover, it is important to note that although this metric improved with the design of the AQT 2.0, it was the inclusion of user profiling that continued to show improvement over and above the original AQT 1.0. This result was mainly due to the decrease in the number of paths owing to the utilization of user profiling.

The values for *Application Flexibility* (**DFA**) and *Dialog Flexibility* (**DFD**) are not statistically significant, since the values obtained were not higher than the threshold of 15 that was set by Rauterberg [RAUTE97]. The threshold of 15 was set by Rauterberg because they could not find significant performance differences for dialog structures below this threshold.

4.3 Summary of Results

By applying our methodology to a test case, we have augmented a virtual distributed library with user profiling and quantitatively verified the results in changes to the user interface using a suite of metrics. This chapter provides the resulting user interface and subsequent product-oriented usability analysis. The

research shows that user profiling is a viable method for augmenting virtual distributed library user interfaces. The slight decrease in *Functional Feedback* is expected due to the increase in parameters having to be adjusted without user profiling. However, the gains in *Interactive Directness* are striking and significant to the overall usability of an interface implementing user profiling.

5 Conclusions

The immense volume of detailed imagery and electrooptical signature data resident throughout the DoD sensor data community has led AFRL/SN to create a Virtual Distributed Laboratory. A promising query interface, called the *Advanced Query Tool,* was designed and quickly prototyped to permit queries of the data organized into remote metadata databases distributed around the DoD. However, the AQT is now in its second iteration of development and needs a technology that allows end users to retrieve their preferences from previous interactive data-inquiry sessions. Furthermore, no analysis had previously been completed to determine the usability of the evolving interface of the AQT. As one promising solution, we submitted that user profiling should be the technology to fulfill the users' need for retrieving past preferences in their signature data inquiry sessions. Moreover, we proposed product-oriented usability analysis metrics that could be used to quantitatively verify the usability of the augmented interface in development. Consequently, the two goals of this research effort were to implement user profiling into the AQT as a proof-of-concept demonstration and complete a product-oriented usability analysis.

The significant contribution of this research is the introduction of a design methodology that incorporates user profiling concepts into a virtual distributed

library and subsequently quantitatively verifies its inclusion as a viable solution. This two-step methodology was successfully demonstrated using AFRL/SN's VDL *Advanced Query Tool* 1.0 and 2.0.

5.1 Conclusions

As a result of the knowledge obtained from the research discussed in the previous chapters, we draw the following conclusions about user profiling and usability analysis in Sections 5.1.1 and 5.1.2 respectively.

5.1.1 User Profiling

Section 4.1 demonstrates an augmented AQT 2.0 via the inclusion of server-side caching of user profiles. User profiling is a mature technology described in current research literature and detailed in Section 2.2. We were able to integrate user profiling into the AQT 2.0 and successfully extract and record users' preferences during interactive metadata query sessions. This will benefit customers of the AFRL/SN VDL who intend to run numerous queries against the available remote data sources. Section 5.1.2 explains the application of a metrics suite to verify the usability of this newly augmented interface.

5.1.2 Interface Usability Analysis

In Section 2.4, we introduce a series of metrics to permit the quantitative

corroboration of the usability of an interface. Section 4.2 provides our analysis of the various iterations of the *Advanced Query Tool* using these metrics. The *Interactive Directness* (**ID**) metric provides the best evaluation of usability of a graphical user interface because it computes a measure of how many dialogs a user must traverse before accomplishing the desired undertaking. The *Functional Feedback* (**FB**) metric shows little variation due to the usage of a graphical user interface for all three versions of the AQT and the heavy weighting interface-type has in the final calculation of this metric. Furthermore, both *Application Flexibility* (**DFA**) and *Dialog Flexibility* (**DFD**) are statistically insignificant because of the similarity of AQT dialog structures.

5.2 Future Areas of Research

There are two areas of related research that are relevant to improving the query interface to virtual distributed libraries. The first area is the addition of central host user profiling to maximize the capability provided by user profiling. The second area is the improvement of the interface's usability.

5.2.1 Central Host User Profiling

There are many benefits to be gained by pushing certain information (i.e. user information and metadata of query result sets) to a central host server, as discussed in Section 3.1.3. These benefits include central reporting capabilities

and the ability to identify underlying trends that prescribe critical future electrooptical signature data requirements. The ultimate management functionality in the business world is a "pay-per-byte" service that bills customers according to their usage. A highly publicized, centrally-controlled, user-friendly virtual distributed library provides that capability.

5.2.2 Interface Usability Improvement

Another research area is that of interface usability improvements that may be quantitatively verified by the methodology we have introduced in Section 3.2. Client-side caching, as discussed in Section 3.1.2, could be implemented to allow users to modify their own profiles offline, if the security requirements are applied to the change in design of the VDL. Another possibility is to display the results of queries graphically, in a manner that makes easier the comprehension of a query's result. Also, automatic profiling in which artificial intelligence is used to select the user's most likely choices should be considered. Finally, research should be accomplished to verify that the user interface of the AQT is compatible with all commonly known web browsers.

5.3 Summary

While the user profiling techniques and subsequent usability analysis applied in this research have provided a contribution to an evolving virtual

distributed library query interface, the constant growth of distributed data holdings will continue to challenge researchers to find solutions that maximize interface usability. We have introduced a methodology for improving digital library interfaces utilizing user profiling techniques and measuring improvements by means of a set of usability analysis tools.

Appendix A – Source Code Availability

The source code used in this research is available by contacting the AFIT, School of Engineering and Management, Database Systems, Point of Contact (POC). Currently, the Database Research POC is:

Maj Michael L. Talbert
Air Force Institute of Technology
WPAFB, OH 45433-7765
E-mail: michael.talbert@afit.af.mil
Phone: (DSN) 785-6565 x4280 Commercial: (937) 255-6565 x4280

Appendix B – Default.cfg (Original Default Profile)

```
1
sdms.SAR_VIEW
24
COLLECTION.SAR_VIEW,COLLECTION,list,null,true,false,MAD 98 Site 1 Scene 1,MAD
98 Site 11 Scene 1,MAD 98 Site 11 Scene 2,MAD 98 Site 11 Scene 3,MAD 98 Site 12
Scene 1,MAD 98 Site 12 Scene 2,MAD 98 Site 13 Scene 2,MAD 98 Site 13 Scene
3,MAD 98 Site 6 Scene 1,MAD 98 Site 9 Scene 1,MSTAR Collection 1 Scene 1,MSTAR
Collection 1 Scene 2 Subscene 1,MSTAR Collection 1 Scene 2 Subscene 2,MSTAR
Collection 1 Scene 2 Subscene 3,MSTAR Collection 1 Scene 3,MSTAR Collection 1
Scene 4,MSTAR Collection 2 Scene 1 Subscene 1,MSTAR Collection 2 Scene 1
Subscene 2,MSTAR Collection 2 Scene 1 Subscene 3,MSTAR Collection 2 Scene 1
Subscene 4,MSTAR Collection 2 Scene 1 Subscene 5,MSTAR Collection 2 Scene
2,MSTAR Collection 2 Scene 3,MSTAR Collection 2 Scene 4,MSTAR Collection 3
Scene 1,MSTAR Collection 3 Scene 2,MSTAR Collection Squint
COLLECTION_DATE.SAR_VIEW,COLLECTION_DATE,list,null,true,false,19950825,19950831
,19950901,19950902,19950903,19950904,19950905,19961119,19961120,19961121,199611
22,19961123,19970320,19970321,19970508,19970509,19970510,19970511,19970513,1998
0427,19980504,19980506,19980508
DEPRESSION_ANGLE.SAR_VIEW,DEPRESSION_ANGLE,minmax,null,true,false,9,49.88
RANGE_RESOLUTION.SAR_VIEW,RANGE_RESOLUTION,minmax,null,true,false,0.3047,1
CROSS_RANGE_RESOLUTION.SAR_VIEW,CROSS_RANGE_RESOLUTION,minmax,null,true,false,0
.3047,1
RADAR_MODE.SAR_VIEW,RADAR_MODE,list,null,true,false,Spotlight,Spotlite
SITE.SAR_VIEW,SITE,list,null,true,false,eglin_fl,prv_nm,redstn,thr_nm
SITEVAL.SAR_VIEW,SITEVAL,list,null,true,false,Eglin AFB, FL,Estancia,
NM,Redstone Arsenal, AL,SomeWhere, NM
SQUINT_DIR.SAR_VIEW,SQUINT_DIR,list,null,true,false,Left,Right
SPLANE_SQUINT.SAR_VIEW,SPLANE_SQUINT,minmax,null,true,false,-48.8228,47.7977
OBJ_TYPE_ID.SAR_VIEW,OBJ_TYPE_ID,list,null,true,false,240MMMRL,2S1,BMP1,BMP2,BR
DM2,BTR60,BTR70,BTR80,BVP80,D7,GAZ66,HEMTT,HMMWV,LARGE
SURROGATE,M1,M1002,M109,M110,M113,M2,M3,M35,M520,M548,M577,M60,M88,M911,M978,MT
U20,SA13 TELAR,SA6 SPU,SA8 SPU,SA8
TZM,SCUD,SLICEY,T62,T72,URAL4320,ZIL131,ZSU23-4,null
OBJ_SERIAL_NBR.SAR_VIEW,OBJ_SERIAL_NBR,list,null,true,false,077L,0AP00N,1,132,1
39A,191-
1310,196BDEF1181,222RF,262A,264A,266A,268A,274A,276A,3336,462L,502020,568L,5843
95,6705,6706,699,7510TH,812,83A,92V13015,9563,9566,A04,A05,A06,A07,A10,A32,A50,
A51,A52,A54,A58,A60,A62,A63,A64,B01,B28,B88,C-17,C-
27,C11,C13,C21,C23,C245HAB,C40,C56,C58,C62,C69,C70,C71,C72,C73,C83,C84,D06,D08,
D12,D30,E-11,E-71,E11,E12,E38,E40,E41,E72,F12,F20,F40,H1,HQ-61,HQ-
71,HV984,K10YT7532,K113,M06,M08,MV02FP,MV02GX,NPOGSN,R23,R26,R80,S7,T04,T505,T8
39,TE002A,null
OBJ_TYPE_SER.SAR_VIEW,OBJ_TYPE_SER,list,null,true,false,-,240MMMRL-TE002A,2S1-
B01,BMP1-C11,BMP1-C13,BMP2-9563,BMP2-9566,BMP2-C21,BMP2-C23,BRDM2-E-71,BRDM2-
E72,BTR60-C62,BTR60-C69,BTR60-K10YT7532,BTR70-C70,BTR70-C71,BTR70-C72,BTR70-
C73,BTR80-R80,BVP80-C40,D7-92V13015,GAZ66-E38,HEMTT-222RF,HEMTT-NPOGSN,HMMWV-
HV984,HMMWV-T04,LARGE SURROGATE-6705,LARGE SURROGATE-6706,M1-0AP00N,M1-
584395,M1002-C83,M1002-C84,M109-A58,M109-A60,M109-B28,M109-B88,M109-C56,M109-
C58,M110-196BDEF1181,M110-H1,M113-077L,M113-462L,M113-568L,M113-699,M113-C-
17,M113-C-27,M2-MV02FP,M2-MV02GX,M2-T505,M3-E-11,M35-139A,M35-262A,M35-
264A,M35-266A,M35-268A,M35-274A,M35-276A,M35-83A,M35-K113,M35-T839,M520-191-
1310,M548-C245HAB,M577-HQ-61,M577-HQ-71,M60-3336,M88-502020,M911-7510TH,M978-
NPOGSN,MTU20-A50,SA13 TELAR-D30,SA6 SPU-M06,SA8 SPU-M08,SA8 TZM-D12,SCUD-
```

```
R23,SCUD-R26,SLICEY-1,T62-A51,T62-A52,T62-A54,T72-132,T72-812,T72-A04,T72-
A05,T72-A06,T72-A07,T72-A10,T72-A32,T72-A62,T72-A63,T72-A64,T72-S7,URAL4320-
E40,URAL4320-E41,ZIL131-E11,ZIL131-E12,ZIL131-F12,ZIL131-F20,ZIL131-F40,ZSU23-
4-D06,ZSU23-4-D08
DEG_ASPECT_ANGLE.SAR_VIEW,DEG_ASPECT_ANGLE,minmax,null,true,false,-
179.9930,179.9950
TURRET_ROTATION.SAR_VIEW,TURRET_ROTATION,minmax,null,true,false,0,355
BARREL_ELEVATION.SAR_VIEW,BARREL_ELEVATION,minmax,null,true,false,-9,45
DOOR_HATCH.SAR_VIEW,DOOR_HATCH,list,null,true,false,none,some,null
OTHER_ARTICULATION.SAR_VIEW,OTHER_ARTICULATION,list,null,true,false,no,yes,null
CONFIGURATION.SAR_VIEW,CONFIGURATION,list,null,true,false,normal,variant,null
ACCESSORIES.SAR_VIEW,ACCESSORIES,list,null,true,false,none,null
CAMOUFLAGE.SAR_VIEW,CAMOUFLAGE,list,null,true,false,1 radar scattering net,
over top and sides, elevated off vehicle,back 50% of vehicle is covered with 1
radar scattering net,none,null
CAMO_PERCENT.SAR_VIEW,CAMO_PERCENT,minmax,null,true,false,0,50
OBSCURATION.SAR_VIEW,OBSCURATION,list,null,true,false,20 ft reinforced concrete
wall,depth of revetment,none,null
OBS_PERCENT.SAR_VIEW,OBS_PERCENT,minmax,null,true,false,0,87
```

Bibliography

[FOXSO95] E. Fox (ed.), *Source Book on Digital Libraries, TR 93-35,* Virginia Polytechnic Institute and State University, p. 65, 1993.

[FRANK96] M. Franklin, S. Zdonik, "Dissemination-Based Information Systems," In *IEEE Data Engineering Bulletin,* pp. 19-28, September 1996.

[FRANK97] M. Franklin, S. Zdonik, "A Framework for Scaleable Dissemination-Based Systems," In *Proceedings of the 1997 ACM SIGPLAN Conference on Object Oriented Programming Languages and Applications (OOPSLA'97),* pp. 94-105, October 1997.

[FRENC99] J. French, C. Viles, "Personalized Information Environments," In *D-Lib Magazine '99,* June 1999.

[GAUCH94] S. Gauch, R. Aust, J. Evans, J. Gauch, G. Minden, D. Niehaus, J. Roberts, "The Digital Video Library System: Vision and Design," In *Digital Libraries '94,* 1994.

[GLADN96] H. Gladney, Z. Ahmed, R. Ashany, N. Belkin, E. Fox, M. Zemankova, "Digital Library: Gross Structure and Requirements (Report from a Workshop)," *IBM Research Report RJ 9840,* pp. 1-20, May 1994.

[HARVE99] L. Harvey, R. Marshak, "Building Valuable Customer Relationships: Knowing Your Customers Better Through Interaction," Prepared for the NCR Corporation by the Patricia Seybold Group, Boston, 1999.

[JANNI96] J. Jannink, D. Lam, N. Shivakumar, J. Widom, D. Cox, "Data Managment for User Profiles in Wireless Communication Systems," Prepared for the Computer Science & Electrical Engineering Departments, Stanford University, Stanford, 1996.

[KURZK98] C. Kurzke, M. Galle, M. Bathelt, "WebAssist: A User Profile Specific Information Retrieval Asssistant," *Computer Networks and ISDN Systems,* pp. 654-655, September 1998.

[LUEIC98] Q. Lu, M. Eichstaedt, D. Ford, "Efficient Profile Matching for Large Scale Webcasting," *Computer Networks and ISDN Systems*, pp. 443-455, April 1998.

[LEINE98] B. Leiner, "The Scope of the Digital Library," Prepared for the *DLib Working Group on Digital Library Metrics*, October 1998.

[MCKIE99] S. McKie, "Powering Better Customer Relationships from the Back Office," In *Intelligent Enterprise*, pp. 27-31, July 1999.

[MAHAJ96] R. Mahajan, B. Shneiderman, "Visual & Textual Consistency Checking Tools for Graphical User Interfaces," *Technical Report CS-TR*-3639, University of Maryland, 1996.

[MANJU95] B. Manjunath, "Image Browsing in the Alexandria Digital Library (ADL) Project," In *D-Lib Magazine '95*, August 1995.

[MUELL99] B. Mueller, "Keeping Your Customers from Defecting," In *Beyond Computing*, Vol. 8, No. 3, April 1999.

[RAUTE92] M. Rauterberg, "A Product Oriented Approach to Quantify Usability Attributes and the Interactive Quality of User Interfaces," In *Work With Display Units '92*, Elsevier Science Publishers B. V., 1992, pp. 324-328.

[RAUTE95] M. Rauterberg, "Four Different Measures to Quantify Three Usability Attributes: 'Feedback', 'Interactive Directness', and 'Flexibility'," In *Design, Specification, and Verification of Interactive Systems '95*, New York: Springer, 1995, pp. 209-223.

[RAUTE97] M. Rauterberg, "About a Method to Measure the Ergonomic Quality of User Interfaces in a Task Independent Way," In *Workshop on Emerging Technologies in Human Engineering Testing and Evaluation, NATO Research Study Group 24, Human Engineering Testing and Evaluation*, Brussels, 1997.

[SHNEI94] B. Shneiderman, A. Rosenfeld, G. Marchioni, W. Holliday, G. Ricart, C. Faloutsos, J. Dick, "QUEST—Query Environment for Science Teaching," In *Digital Libraries '94*, 1994.

[SHNEI97] B. Shneiderman, D. Byrd, B. Croft, "Clarifying Search: A User-Interface Framework for Text Searches," In *D-Lib Magazine '97*,

January 1997.

[SPINK98] A. Spink, T. Wilson, D. Ellis, N. Ford, "Modeling Users' Successive Searches in Digital Environments," In *D-Lib Magazine '98*, April 1998.

[STRAT99] P. Stratton, "A Metrics-based Analysis of Interface Usability Improvements by Applying Intelligent Agents," *Master's Thesis: AFIT/GCS/ENG/99M-18*, Air Force Institute of Technology, March 1999.

[VANHO96] N. Van House, M. Butler, V. Ogle, L. Schiff, "User-Centered Iterative Design for Digital Libraries," In *D-Lib Magazine '96*, February 1996.

[VDLFO99] "The VDL for ATR Mission Statement," Available at http://www.standevalexp.vdl-atr.afrl.af.mil/mission/index.htm, 1999.

[YANAN95] T.W. Yan, J. Annevelink, "A Powerful Wide-Area Information Client," In *Proceedings of the 1995 IEEE Computer Conference (COMPCON'95)*, pp. 13-18, March 1995.

[YANGA94] T.W. Yan, H. Garcia_Molina, "Distributive Selective Dissemination of Information," In *Proceedings of the Third International Conference on Parallel and Distributed Information Systems (PDIS'95)*, pp. 89-98, September 1994.

[YANGA95] T.W. Yan, H. Garcia_Molina, "SIFT – A Tool for Wide-Area Information Dissemination," In *Proceedings of the 1995 USENIX Technical Conference*, pp. 177-186, January 1995.

Vita

Captain Jason T. Ward was born on 23 November 1973 in Abilene, Texas. In 1991, he graduated as salutatorian from Waynesboro Area High School, Waynesboro, Pennsylvania. Upon graduation, he entered and became an academically distinguished graduate of the Class of 1995 at the United States Air Force Academy. Later in 1995 he attended Basic Communications Officer Training at Keesler AFB, Mississippi. After completion of this training, he completed his first tour of duty at Ramstein AB, Germany, as Chief of Software Engineering and Development for the USAFE Computer Systems Squadron. While stationed at Ramstein, he deployed to Vicenza, Italy, in March 1998 to spend four months as the Bosnia Command and Control Augmentation Systems Officer-In-Charge. In August 1998, he was re-assigned to the Air Force Institute of Technology at Wright-Patterson AFB, Ohio. Upon graduation in March 2000, Capt Ward will be assigned to the Air Force Research Laboratory, Focused Energy Directorate, at Kirtland AFB, New Mexico.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2000	Master's Thesis

4. TITLE AND SUBTITLE

ENHANCING A VIRTUAL DISTRIBUTED LIBRARY USER INTERFACE VIA SERVER-SIDE USER PROFILE CACHING

5. FUNDING NUMBERS

EN #99-308

6. AUTHOR(S)

Jason T. Ward, Captain, USAF

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/00M-23

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Virtual Distributed Laboratory
AFRL/SNAS
Attn: Wilson, T.A.
2010 Fifth St.
WPAFB, OH 45433 DSN: 785-6329 x2730

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Maj. Michael L. Talbert, AFIT/ENG, DSN: 785-6565 x4280

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; Distribution Unlimited

12b. DISTRIBUTION CODE

ABSTRACT (*Maximum 200 Words*)

Various Department of Defense (DoD) agencies archive terabytes of intelligence imagery and electrooptical signature data. The Air Force Research Laboratory, Sensors Directorate (AFRL/SN), is tasked with creating and managing a virtual distributed library that facilitates secure, detailed queries across these distributed holdings using the internally developed *Advanced Query Tool* (AQT). In this research, a methodology is proposed to utilize user profiling techniques to augment a digital library. As part of this methodology, product-oriented usability analysis metrics are introduced that quantitatively verify the usability of an interface. The methodology is applied to the AFRL/SN's Virtual Distributed Laboratory AQT and subsequently analyzed using the suite of product-oriented usability metrics. The results of applying the methodology to the test case demonstrated significant, quantitatively verifiable improvements to the AQT interface. This research establishes that user profiling may be utilized to greatly improve the user interface of a digital library.

14. SUBJECT TERMS	15. NUMBER OF PAGES
User Profiling, User Interface Metrics, Digital Library, Query Interface	87
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102